Project_2

Oliver Tausendschön, Manuel Carli, Caroline Meyer, Benedek Takacs

2023-11-15

Contents

Report

Introduction / Abstract

In this Report, we take a deep dive into the world of customer reviews to uncover trends and insights related to a selected product on Amazon.com. With a foundation of about 500 text-based evaluations, the product that we have carefully picked from a product category of our choosing offers big variety of thoughts and comments.

The R code of the entire process is available as an appendix, revealing the details of our analysis procedure. Additionally, our presentation summarises the most significant discoveries, providing a quick overview of the brand's online presence and consumer attitude. The gathered dataset, in .rda format provides the basis for our investigation and creates a clear picture of the brand's interaction with its clientele.

Our Aims

Our goal is to analyse and extract useful data from the collection of customer reviews. In doing this, we want to provide basic answers that explain the brand's effectiveness and consumer impression.

In order to get a good impression of customer´s opinions, we will analyse common and frequent terms that appear in the reviews of this data. We will also explore temporal dimension, tracking indicators as they change over time to provide insights how the product or the opinions evolve over time. We will be examining the tone of the text to determine the emotional undertones that are typical of customer feedback. Finally, we will interpret the subjects that predominate in customer conversations, revealing the problems and elements that are most important to the clientele.

Our ultimate objective is to offer the brand with insights that can be put into practise as we go through these elements. If the brand's goals are a high star rating and happy consumers, our study will help them in archieving that.

Preparation

Before analysing the data, we actually have to get the data. We did this by using a basic webscraper to extract important information on the prodct´s review page.

Scraping the data

To scrape reviews which we can then analyze we start by installing necessary / helpful packages. We could then start with tasks such as HTML parsing and mimicking a browser with a machine.

However, we quickly ran into problems as Amazon makes an effort to prevent scraping, particularly when many pages are being scraped. We started by trying the code with different products and pages, experimenting with differend HTML nodes and xpaths. All this was done with the R-package Rvest. In the end, we created custom headers, a random time out to to miick human behaviour as well as to avoid being "too aggressive" and imitated request headers. We managed to scrape 100 reviews without filtering for specific terms. More reviews where not attainable, since Amazon, as part of their web-scraping prevetion, limits the number of pages of reviews to be viewed to 10, and the number of reviews per page also to 10. To increase the dataset size, we filtered for all five possible star ratings and obtained the 100 available reviews per star rating, resulting in a total of 500 (5x100) reviews. However, this comes with the big downside of introducing a bias. We nevertheless decided to continue as one purpose of this project is to carry out a complete analysis and larger, although biased, dataset allows for intersting insights into the different wordings, sentiments and emotions associated with different star ratings.

We decided to extract information on the title of the reviews, the review content, the review date, whether the review is verified or not, how many people found it helpful, and the star rating. Our dataset thus contains 500 observations with 6 variables.

The variables are the following:

- `review_title`: This is the title of the review
- 'review_text": This is the review itself.
- `formatted_date`: This is the date when the review was published.
- `verified`: This indicated whether a review is made by an officially verified buyer or not.
- `N_helpful`: This is the count of people that marked the review as being helpful.
- `star_rating_num`: This is the score of the review. The maximum is five and minimum is zero. A higher number corresponds to a more positive opinion about the product.

The technical details of the data analysis are provided in the appendix, accompanied by the corresponding R code.

General Analysis

To get an idea of the data we are working with, We want to start by taking a look at some general features of our review data. We will keep this section short as it only serves to get a basic understanding of the data.

Let us start by taking a look at **when the reviews were made**: The time frame of our reviews spans from 06th of June, 2020, when the first review was published until the 17th of November, 2023, where the most recent one was written. This shows and ensures that we consider reviews in a long time span, considering potential product updates, relaunches and alike.

It is also interesting to get more of an insight into the **general rating behavior**. The **worst rating** is 1 stars, the **highest one** is 5 stars and the **average rating** amounts to 3 stars. This can be shown by using a histogram of the distribution of star ratings. Note that the equal distribution is due to the way our webscraper works.

Another visualization of the **distribution of star ratings** might be helpful to see how the product has performed in general. Thus, we present a histogram:

We can also visalize the average star rating across time. This gives us a good guess of how the quality of the product might deteriorate or improve.

It seems as the average rating improves in the long term, which is a good sign. Let´s see if there is any correlation with **the number of reviews published across our time period**.

Also, the average rating seems to improve already after the first reviews and stays relatively constant throughout the time period where reviews were published.

It might also be interesting to take a look at **the number of reviews published across our time period** and see if and how this corresponds to the development of the star ratings.

We can see a bit of seasonality in this plot but mainly, there is an increasing tendency of the reviews.

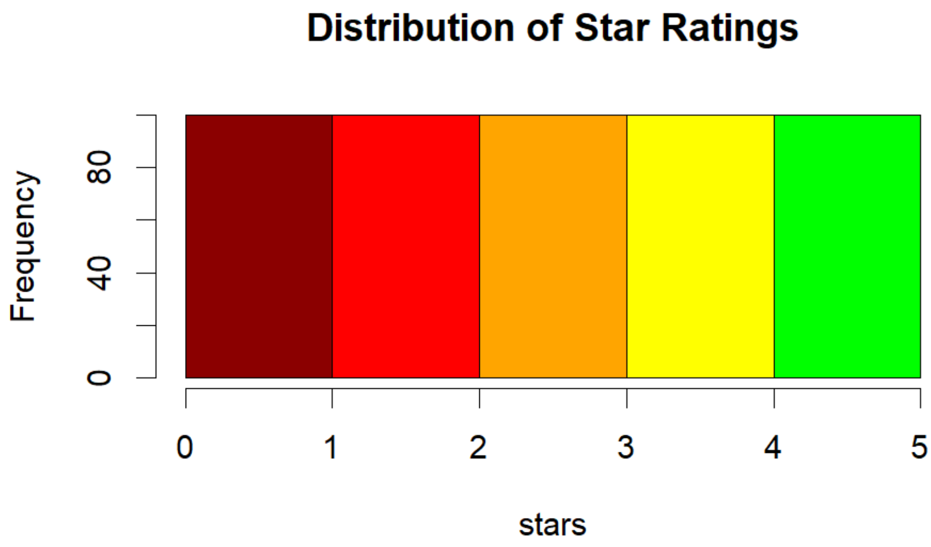


Figure 1: Distribution_of_Stars.

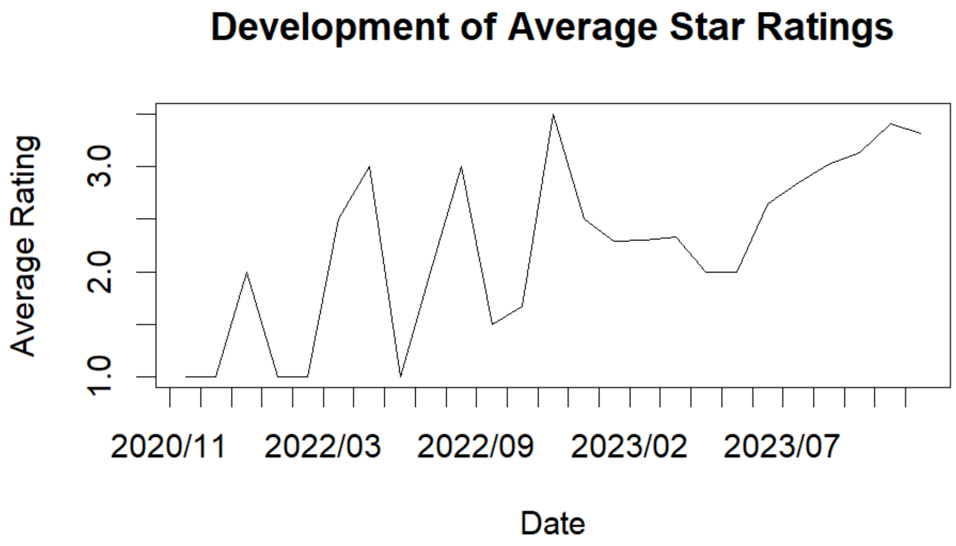


Figure 2: Development_Stars.

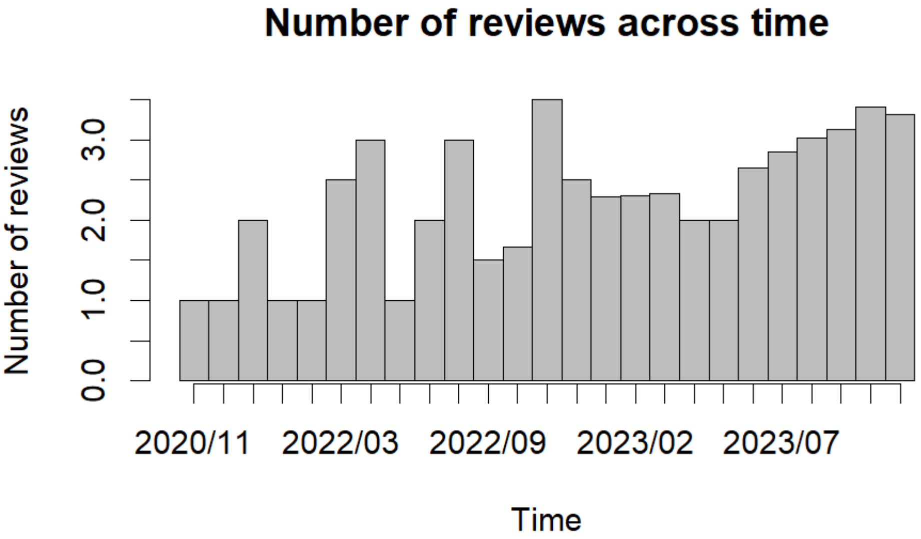


Figure 3: Number_of_reviews.

When comparing the two plots we can see that overall the review-situation improved over time: Both more reviews and better reviews were published in the long run. The dip in June might be interesting to analyze - especially to see if it remains so apparent when the dataset of reviews is larger and less restricted.

Furthermore, we might want to take a look at **helpfulness of the reviews**. We can see that the reviews were rated as helpful by between 0 and 50 people, with an average of 3.47 and a median of 0 as well. This indicates skewedness and strong outliers, so we want to take a closer look:

This supports our hypothesis that the vast majority of reviews are rated as non-helpful by very few to none people. Only very few people attained over 15 "helpfulness"-votes. We can see the "Winner takes it all" principle. Overall a third of the reviews were perceived as helpful, meaning that at least 1 customer rated them as such.

Text Analysis

This section focuses on text analysis, aiming to uncover patterns, relationships, and insights embedded in the textual feedback provided by customers.

To kick off our text analysis, we start by taking a look at review length. We can see that the average length of a review in our dataset is about 70 words. But what might be more interesting is the **correlation between the length of a reviews and its rating. The correlation is about -0.3 What this reveals is that long reviews tend to be more negative or that customers who have a negative opinion about a product tend to share more information to explain or complain.

**correlation between the length of a review and its helpfulness The correlation is at about 0.63 This leads to the conclusion that longer reviews tend to be rated as more helpful as they for example share more information and details that can be useful for other potential customers.

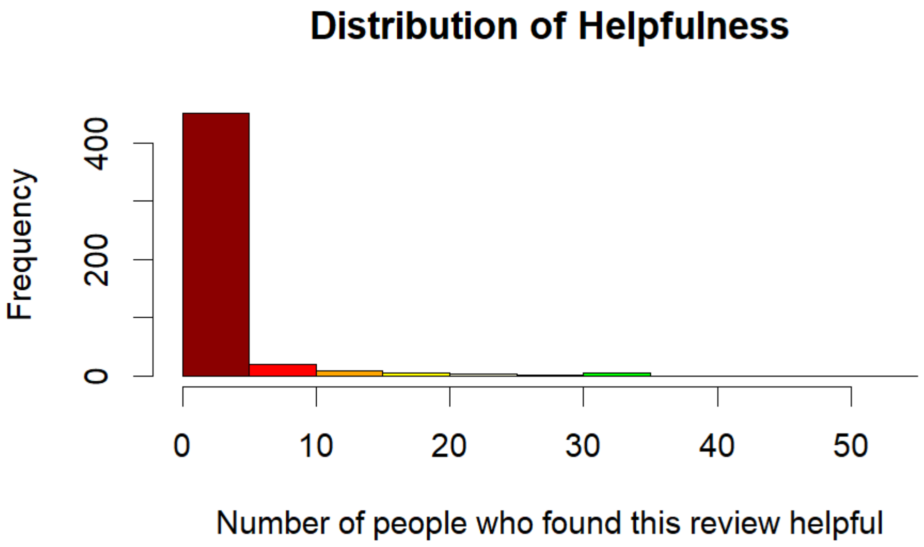


Figure 4: Distribution_of_helpfulness

Now let us dive even deeper into text analysis than merely the word count. Let's have a look at **what people actually write in their reviews**. To do this, we cleaned the reviews by removing special characters, nubers, punctuation, extra white spaces, common stopwords and reduced all words to their stem. After unsurprisingly seeing words like Phone, iphone, apple as the most frequent, we removed these and contructed the following wordcloud:

This reveals a much more interesting picture. We chose that all terms that are mentioned at least 5 times in the 100 reviews are shown in the word cloud. The by far biggest and thus most frequently used significant term in reviews is "battery". Additionally screen and scratches as well as condition might be interesting ones to dive into.

Let us now do the same for the titles of the reviews and see if there are differences:

Since this word cloud is less informative, we will include more words (at lower frequency threshold) and see an overwhelming frequency of very positive terms (good, great, perfect). This is a positive indicator.

Sentiment analysis

Before we dive into applying sentiment analysis on our dataset of reviews, we tried to recall the basics of the Sentiment package and its meaning in R. Essentially sentiment analysis works by differentiating between words depending on the sentiment that is attached to them. This is conducted via dictionaries consisting of lists of positive vs. negative words, or lists of more diverse emotions. Packages such as `sentimentr` in R work by scanning the text to see if words in the text match with any dictionary entries. The words are then assigned a value (>0 if the word is located on the positive list, <0 if it is on the negative one) - all values are added together and the average sentiment is determined. Important note: The packages take the words before and after a term into account in order to assess its classification. This way valence shifters or negations can be included.

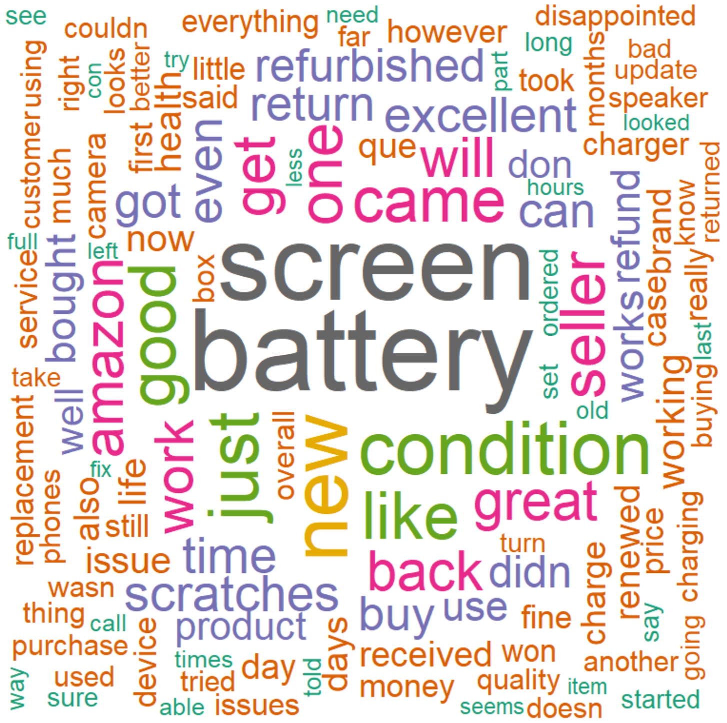


Figure 5: Wordcloud

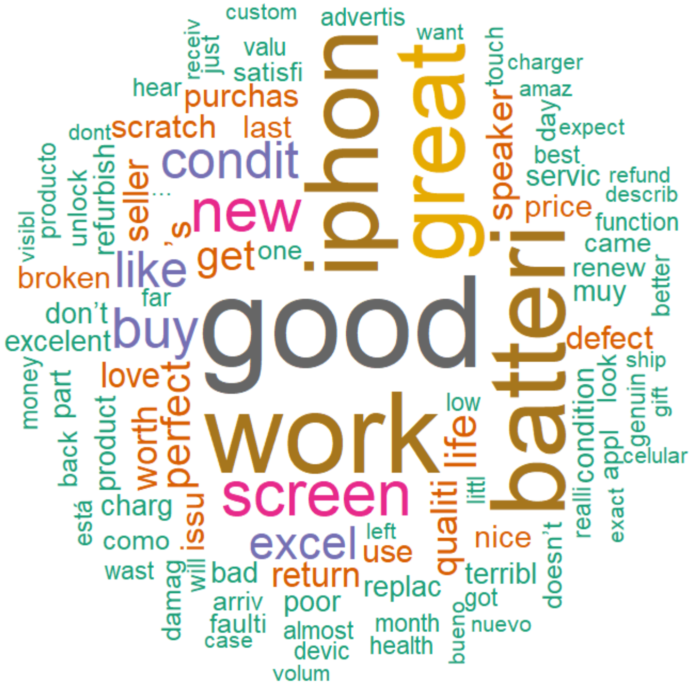


Figure 6: Wordcloud

Sentiment analysis can be applied in a number of fields and situations. Its use-cases range from social media monitoring, political campaigns, PR and market research.

We now want to start our first computations in the field of sentiment analysis to get a better picture about the general tonality and sentiment of the reviews we are examining. In this we will differentiate between title and text look at the correlation between them and their general behavior.

The computations reveal that in the review texts, sentiment ranges from -0.54 to 1.42 and averages at around 0.25. For the titles, the lowest sentiment is -1.44, the highest is 1.23 and the average is around 0.35. Both average sentiments are positive, which is good news for the product. Anything else would be highly unusual since the sentiment should correspond to star ratings and since there are no significant bad ratings, this would need further investigation. We also prove that the sentiment of title and review text are correlated since the p-value is extremely low.

Now, similar to the way we wanted to check if the rating behavior changed over time we want to take a look at the **development of sentiment over time**:

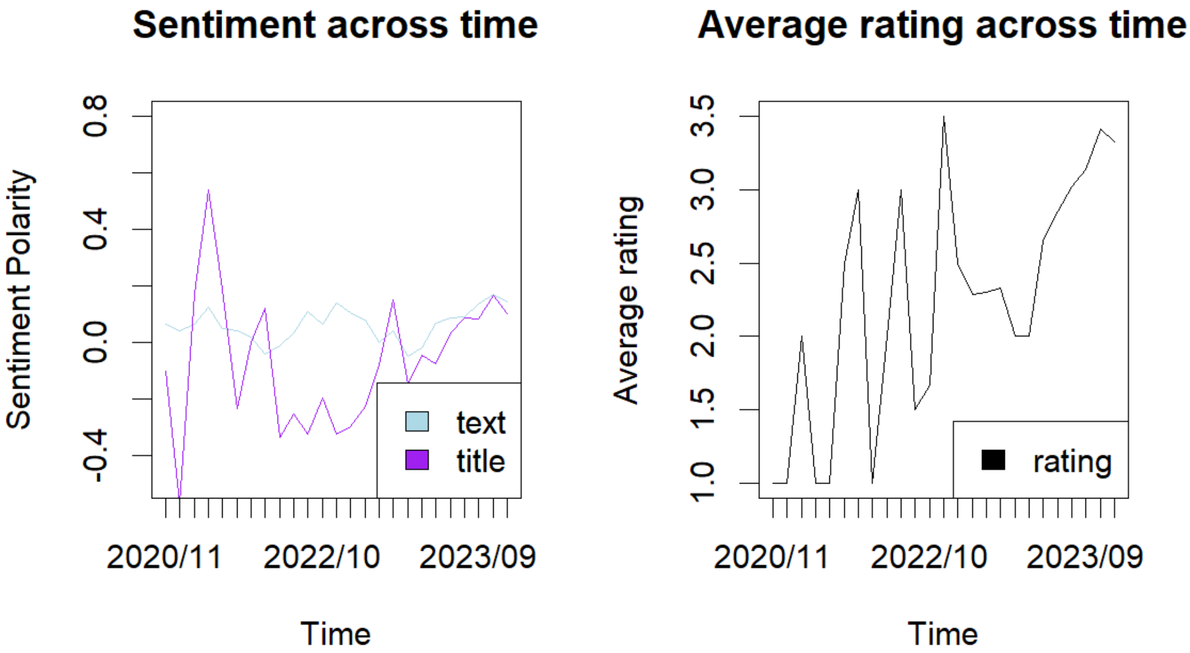


Figure 7: Sentiments

When comparing the plots of how sentiment (in both title and text) developed, we can see a clear correlation with average rating. This is good to see as it confirms that users who put a better rating put more positive emotions into the review.

Let us now analyse how sentiment & other variables interact in more detail:

In this correlation plot/matrix we can see a rather strong positive correlation between sentiment in title and sentiment in text - this we have already found out. Additionally we can see slight negative correlations between text length and sentiment. This further supports our findings that longer reviews will have a worse rating. Star ratings thus negatively correlate with text length, but are

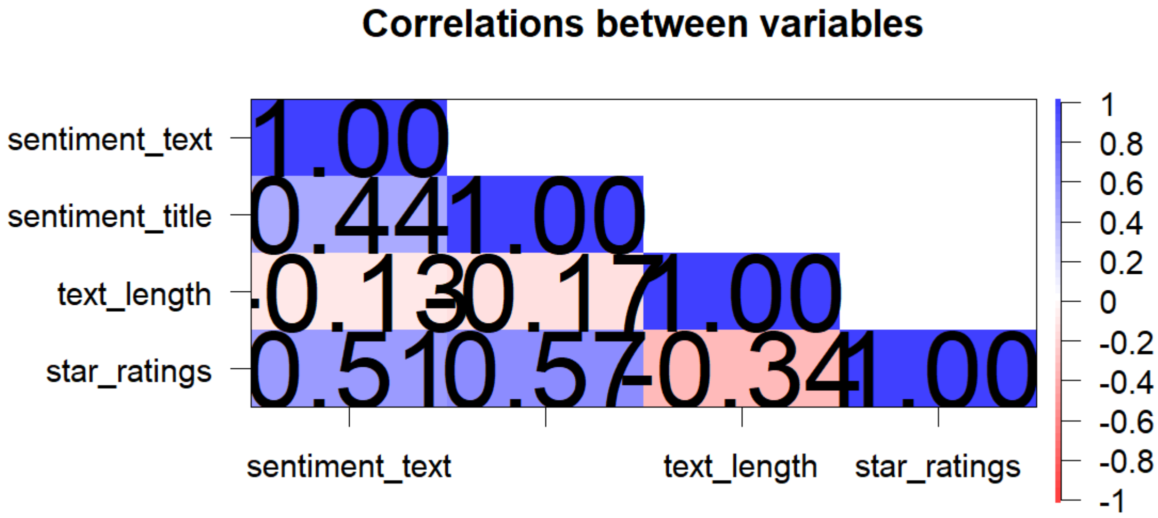
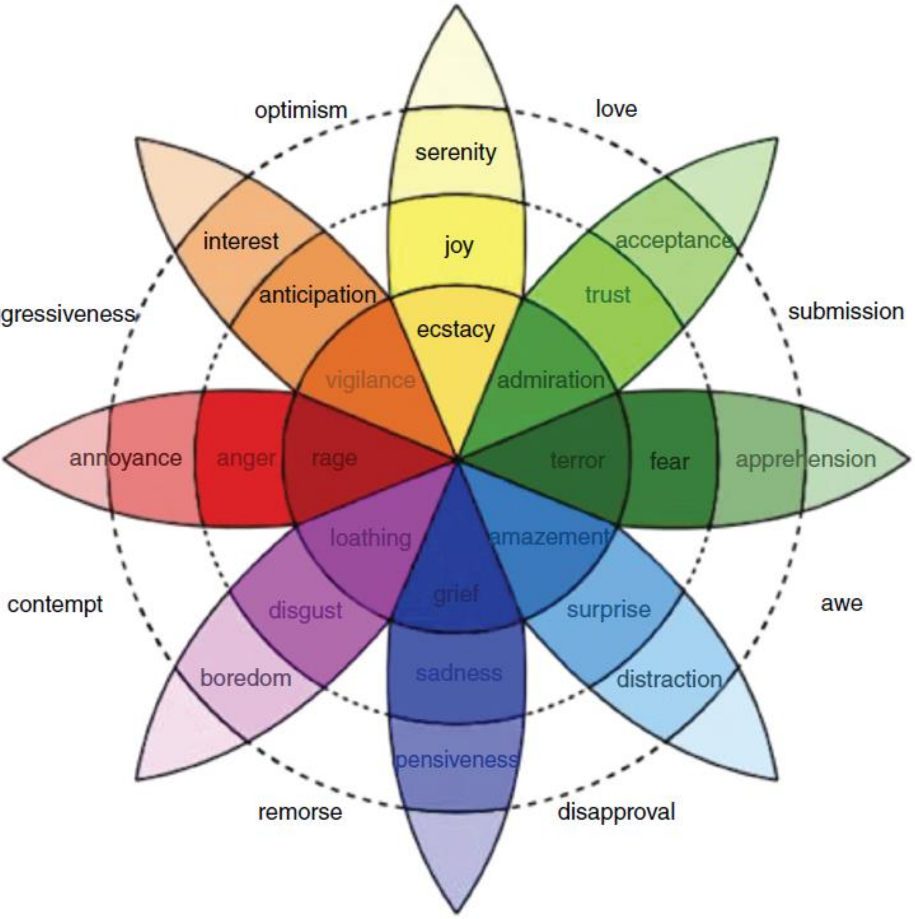


Figure 8: Corplot

positively impacted by the sentiment score. All in all, we are glad to see this as it confirms our previous assumptions.

Emotion Analysis

With emotion analysis we aim to get more detailed insights into the emotions that are expressed in reviews by differentiating not only between positively and negatively connotated terms, but a more developed spectrum of emotions. As a basis, Plutchik's wheel of emotion is used.



{width = 60%}

In order to obtain insights into the emotions in the review texts, we categorize the extracted emotions into the categories of the Wheel of Emotion and examine the degree of average emotion.

Furthermore, in order to determine the relevance of the individual emotions, we examine their prevelance among the review texts:

The barchart above shows the determined prevelance of the indivual emotions. The higher the bar, the higher the relevance of the corresponding emotion. Based on the plot, we can infer the most dominant emotions in the review texts: the latter are anticipation (orange bar), joy (yellow bar) and trust (light green bar). However, also anger plays an important role in the reviews as well as the emotion of surprise.

To increase our understanding of the emotions expressed in the reviews, we chose to examine the correlation between the individual emotions to uncover possible connections and insights into how the different emotions might go hand in hand with another in the reviews we are examining. The therefore obtained results are shown in a correlation matrix.

The results reveal a high positive correlation between joy and anticipation, joy and trust, anticipation and surprise and trust as well as between surprise and anticipation. No noteworthy negative correlation among emotions seems to be present.

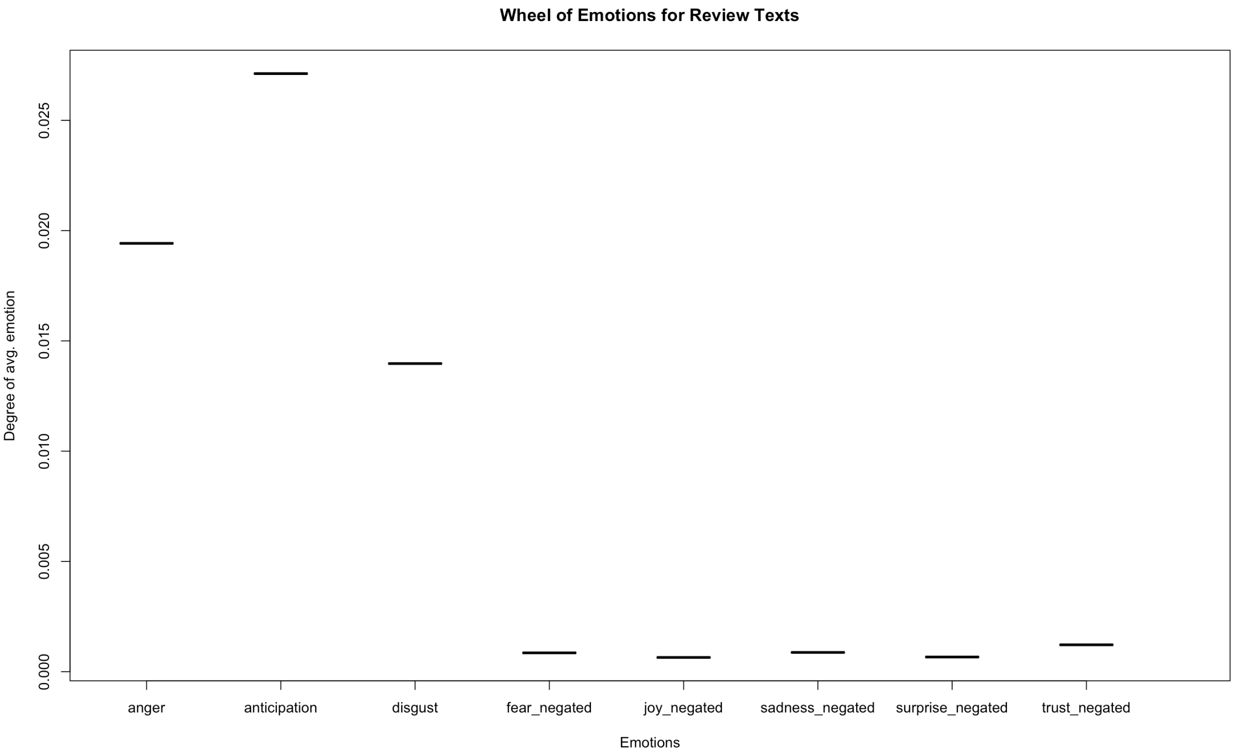


Figure 9: Categorization of review text according to Wheel of Emomotion.

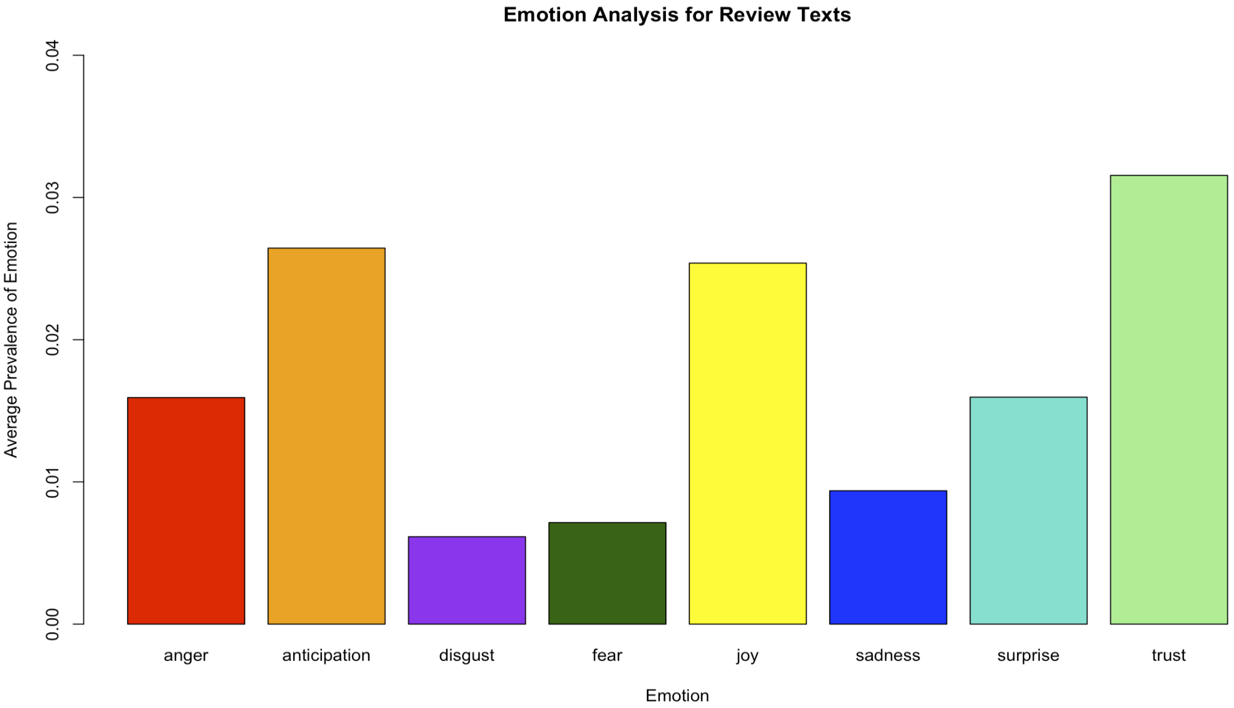


Figure 10: Prevelance of emotions.

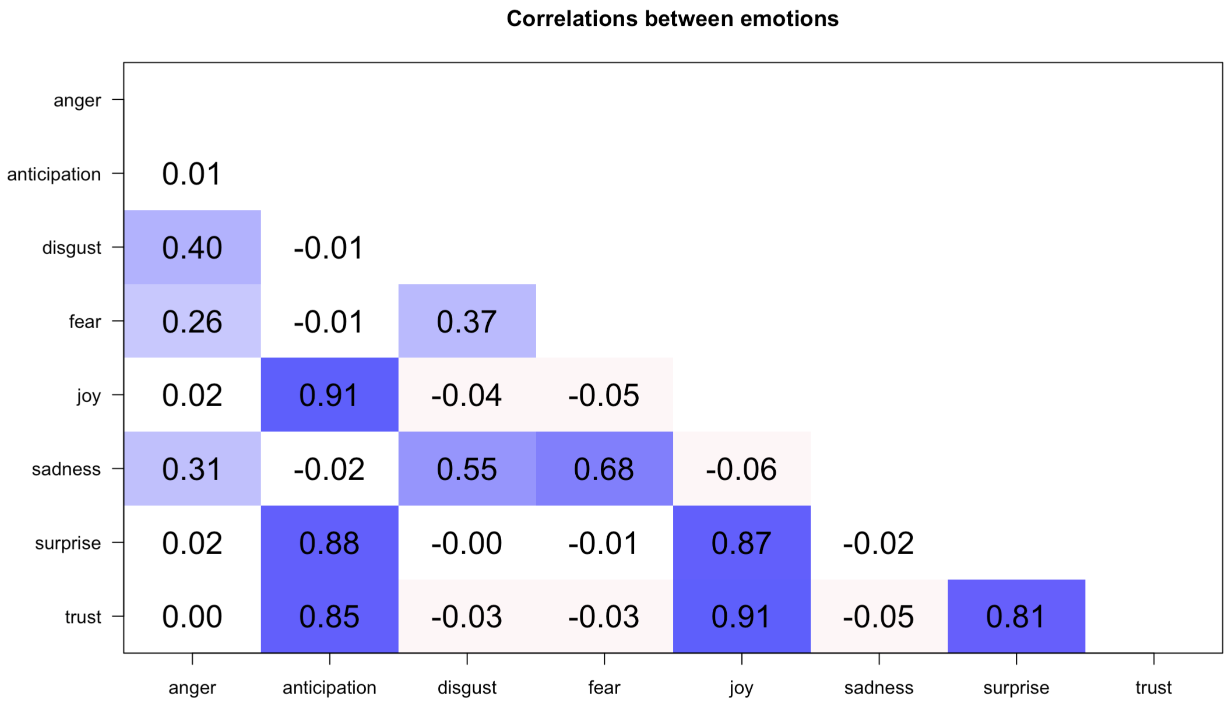


Figure 11: Correlation among different emotions.

Topic Analysis

After getting a feel for the sentiment and emotions present in the reviews for our production under examination, we next set out to analyse the topic of the individual reviews, trying to identify the latent topics among them and to determine what people are talking about. To do this efficiently, we will make us of topic modelling, which refers to methods that aim to identify topics inside a text corpus.

In our case, we use Latent Dirichlet Allocation (LDA), which is a popular probabilistic model in topic modelling. LDA assumes that documents are a mixture of topics and that each word in a document can be attributed to one of the document's topics. To identify latent topics, the model analyses the distribution of words across a collection fo documents.

Process-wise after pre-processing, a document term matrix is created, the number of topics to differentiate between is decided upon, we then check for convergence, estimate the model & can then interpret our findings. The number of topics, which is a key hyperparameter for LDA models, was determined semi-automatically, testing different combinations and choosing the model with the lowest Akaike Information Criterion (AIC). The results suggested that three topics are the optimal choice in our case. (*Note: The detailed procedure can be found in the Appendix.*)

Applying our LDA model, with the number of topics to be determined to three, we identified the following topics:

- Quality: People seem to be talking about the quality of the product and the purchasing process. Words like "good", "work" and "excel" are associated with it.
- Technical Issues: In this case, people seem to be talking about technical issues, mentioning potential issues like "battery", "scratch" and "speaker".

- Value: For this topic, people seem to be adressing the value of the purchased product, which is in our case a refurbished/renewed product. Mentiong word like “great”, “new”, “life”, “condit” and “worth”.

Lastly, we examine the relative importance of the individual topics in our review, which is shown in the chart.

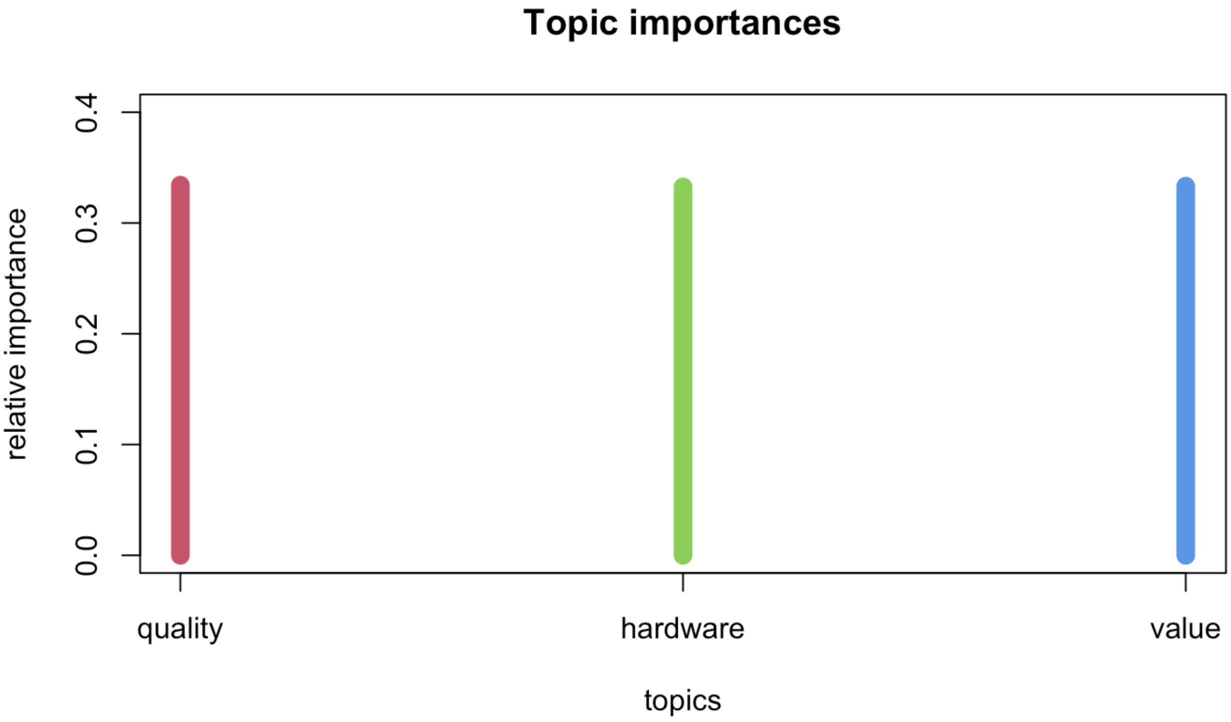


Figure 12: Relevance of individual topics.

Interestingly, all identified topics seem to have the same relative importance. However, one must critically notw that this could also be the case due to the dataset generation process and the hereby introduced bias. Further analysis would be necessary to achieve a clear differentation.

Appendix

The appendix contains the technical analysis of the available data, including all the corresponding R-code. For the analysis we used the following R-packages:

Scraping the data

To scrape reviews which we can then analyze we start by installing necessary / helpful packages:

```
if (!require("pacman")) install.packages("pacman")
```

```
## Loading required package: pacman
```

```
pacman::p_load(rvest,
               polite,
               tidyr,
               dplyr,
               ggplot2,
               tibble,
               purrr)
library(tidyverse)
```

```
## -- Attaching core tidyverse packages ------------------------ tidyverse 2.0.0 --
## v forcats   1.0.0     v readr     2.1.4
## v lubridate 1.9.3     v stringr   1.5.0
```

```
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter()         masks stats::filter()
## x readr::guess_encoding() masks rvest::guess_encoding()
## x dplyr::lag()            masks stats::lag()
## i Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to bec
```

```
library(rvest)
library(vctrs)
```

```
##
## Attaching package: 'vctrs'
##
## The following object is masked from 'package:tibble':
##
##     data_frame
##
## The following object is masked from 'package:dplyr':
##
##     data_frame
```

```
library(tm)
```

```
## Loading required package: NLP
##
## Attaching package: 'NLP'
```

```
##
## The following object is masked from 'package:ggplot2':
##
##     annotate
```

```
library(SnowballC)
library(wordcloud)
```

```
## Loading required package: RColorBrewer
```

```
library(RColorBrewer)
library(ggplot2)
library(udpipe)
library(sentimentr)
library(textcat)
library(pscl)
```

```
## Classes and Methods for R developed in the
## Political Science Computational Laboratory
## Department of Political Science
## Stanford University
## Simon Jackman
## hurdle and zeroinfl functions by Achim Zeileis
```

```
library(topicmodels)
library(psych)
```

```
##
## Attaching package: 'psych'
##
## The following objects are masked from 'package:ggplot2':
##
##     %+%, alpha
```

Next up, we start our web-scraping to read information from a chosen website. Here we faced two major difficulties: Amazon blocks attempts of webscraping when iterated over numerous pages to avoid bots - this we countered by making the request with customer headers. Additionally, however - and we did not find a solution to this - only 100 reviews can be obtained via this code since to see further requests Amazon allows only searches for certain terms. Although we could have implemented some terms to search for and add the reviews to our dataset, we did not think that this is a proper solution. Thus we went with only 100 reviews to start with.

```
# Set parameters for scraping reviews
n_reviews <- 100
reviews_per_page <- 10
iters <- ceiling(n_reviews/reviews_per_page)

# Set the base and additional URL parameters
url_p1 <- "https://www.amazon.com/Apple-iPhone-11-128GB-Black/product-reviews/B07ZPKR714/ref=cr
url_p2 <- "?ie=UTF8&reviewerType=all_reviews&pageNumber="
url_p3 <- "&filterByStar="
```

```r
filters <- c("five_star", "four_star", "three_star", "two_star", "one_star")

# Initialize an empty data frame to store all reviews
reviews_all <- data.frame(NULL)
reviews_scraped <- data.frame(NULL)

# Set a seed for reproducibility
set.seed(1479)

# Loop through the iterations to scrape reviews
for (filter in filters) {
  for (i in 1:iters) {
    # Construct the URL for each iteration
    url <- paste0(url_p1, ifelse(i == 1, "prev_1", paste0("next_", i)), url_p2, i, url_p3, fil

    # Set custom headers to mimic a browser request
    headers <- c(
      'User-Agent' = 'Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like
      'Accept' = 'text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,image/apng,*
      # Add more headers as needed
    )

    # Make the request with custom headers
    response <- httr::GET(url, httr::add_headers(.headers=headers))

    # Check if the request was successful (status code 200)
    if (httr::status_code(response) == 200) {
      # Read the HTML content of the page
      page <- read_html(content(response, "text")) #page <- read_html(content(response, "text")

      # Extract Body of 10 reviews per page
      page <- page %>% html_elements(xpath = "/html/body/div[1]/div[2]/div/div[1]/div/div[1]/di

      # Extract review information from the page
      ## Review Title
      review_title <- page %>%
        html_elements(xpath = "//*[@data-hook='review-title']/span[2]") %>%
        html_text() %>%
        str_trim() %>%
        str_remove_all("<.*?>") %>%
        str_replace_all("\\s+", " ")
      ## Review Text
      review_text <- page %>%
        html_elements(xpath = "//*[@data-hook='review-body']") %>%
        html_text() %>%
        str_trim() %>%
        str_remove_all("<.*?>") %>%
        str_replace_all("\\s+", " ")
      ## Review Star Ratings
      star_ratings <- page %>%
        html_elements(xpath = "//*[@data-hook='review-star-rating']") %>%
        html_text()
      ## Review Dates
      review_dates <- page %>%
        html_elements(xpath = "//*[@data-hook='review-date']") %>%
        html_text()

      ## Review Verified Purchase
      reviews <- html_elements(page, xpath = "//*[@data-hook='review']")
      review_verified <- data.frame(Verified = rep(FALSE, length(reviews)))
      j <- 1
      for (element in reviews)
      {if (str_detect(str_squish(html_text(element)), fixed("Verified Purchase"))) {
        review_verified[j, ] <- TRUE
      }
        j <- j+1}

      ## Review n-helpful
      review_n_helpful <- data.frame(N_helpful = rep(NA, length(reviews)))
      k <- 1
      for (element in reviews) {
        review_helpful <- element %>%
          html_text()

        # Define the pattern to match
        pattern <- "((?:\\d+|One) person|\\d+ people) found this helpful"
        match_result <- str_match(review_helpful, pattern)

        if (!is.na(match_result[1, 2])) {
          n_helpful <- match_result[1, 2]

          if (n_helpful == "One person") {
            review_n_helpful[k, ] <- 1
          } else {
            pattern_2 <- "\\b\\d+\\b"
            extracted_number <- str_extract(n_helpful, pattern_2)
            review_n_helpful[k, ] <- as.numeric(extracted_number)
          }
        }

        k <- k + 1
      }

      # Use str_match on the vector review_dates
```

```r
      dates <- str_match(review_dates, "on ([[:alpha:]]+ [0-9]+, [0-9]+)")[, 2]

      # Convert the extracted dates to a standard date format
      formatted_dates <- as.Date(dates, format = "%B %d, %Y", locale = "en")

      # Convert the star ratings to numeric
      pattern_star <- "([0-9]+\\.[0-9]+) out of 5 stars"
      match_result_star <- str_match(star_ratings, pattern_star)[, 2]
      star_rating_num <- as.numeric(match_result_star)

      # Create a data frame with the extracted information
      reviews_comb <- data.frame(review_title, review_text, formatted_dates, review_verified, 

      # Append reviews to the data frame
      reviews_all <- rbind(reviews_all, reviews_comb)
    } else {
      # Print a warning if the request fails
      warning(paste("Request failed with status code:", httr::status_code(response)))
    }
    reviews_scraped <- rbind(reviews_all)
    # Add a random timeout to avoid being too aggressive
    timeout <- runif(1, 5, 10)
    Sys.sleep(timeout)
    print(paste0(filter, " reviews: iteration ", i, "/", iters, " completed."))
  }
}

#save(reviews_scraped, file="Reviews_Scraped_500_v2.rda")
```

Now let us take a look at the dataframe that we have created via the code chunk above and the structure that we are working with:

```r
load("reviews_scraped_500_v2.rda")
head(reviews_scraped)
```

```
##                                    review_title
## 1 Looks brand new and I love it! iPhone 6 to 11
## 2                  Surprisingly a good purchase
## 3                 Like New - Exceptional Value!
## 4                               New iPhone user
## 5 Great phone! Came with 100% battery capacity.
## 6                  Excellent Replacement Phone!
##
## 1 My iPhone 6 died and was only iOS 10 so it was time to get a new phone. After much delibe
## 2
## 3
## 4
## 5
## 6
##   formatted_dates Verified N_helpful star_rating_num
## 1      2023-10-25     TRUE         9               5
## 2      2023-10-23     TRUE        19               5
## 3      2023-10-25     TRUE         3               5
## 4      2023-11-04     TRUE         3               5
## 5      2023-11-12     TRUE         3               5
## 6      2023-10-21     TRUE         2               5
```

```r
str(reviews_scraped)
```

```
## 'data.frame':    500 obs. of  6 variables:
##  $ review_title   : chr  "Looks brand new and I love it! iPhone 6 to 11" "Surprisingly a goo
##  $ review_text    : chr  "My iPhone 6 died and was only iOS 10 so it was time to get a new p
##  $ formatted_dates: Date, format: "2023-10-25" "2023-10-23" ...
##  $ Verified       : logi  TRUE TRUE TRUE TRUE TRUE TRUE ...
##  $ N_helpful      : num  9 19 3 3 3 2 1 5 1 NA ...
##  $ star_rating_num: num  5 5 5 5 5 5 5 5 5 5 ...
```

Our dataframe consists of 100 observations (maximum number of reviews that can be scraped using our algorithm and not specific search words for Amazon) and 6 features. The features are the title of the review (character), the text of the review (character), the date the review was published (Date), whether it was a verified customer or not (TRUE / FALSE), the number of people that found it helpful and the number of stars the product was rated (both numeric).

In the following steps we will analyze these 100 reviews and their content.

General analysis

We want to start by taking a look at some general features of our review data to get a better understanding of the reviews in general.

Let us start by taking a look at **when the reviews were made**:

```r
min(reviews_scraped$formatted_dates)
```

```
## [1] "2020-11-06"
```

```r
max(reviews_scraped$formatted_dates)
```

```
## [1] "2023-11-17"
```

The time frame of our reviews spans from 25th of February, 2023, when the first review was published until the 16th of November, where the most recent one was made.

Additionally, let's check who made the reviewers, or to be exact: **was the reviewer a verified buyer** or not?

```r
table(reviews_scraped$Verified)
```

```
## 
## TRUE 
##  500
```

This reveals that all 100 reviews were made by verified buyers, thus we can neglect this column.

What might be interesting is to have more of an insight into the **general rating behavior**. Since we have, already when scraping, turned the star rating into a numerical variable the following steps can be done quite easily:

```
min(reviews_scraped$star_rating_num)
```

```
## [1] 1
```

```
mean(reviews_scraped$star_rating_num)
```

```
## [1] 3
```

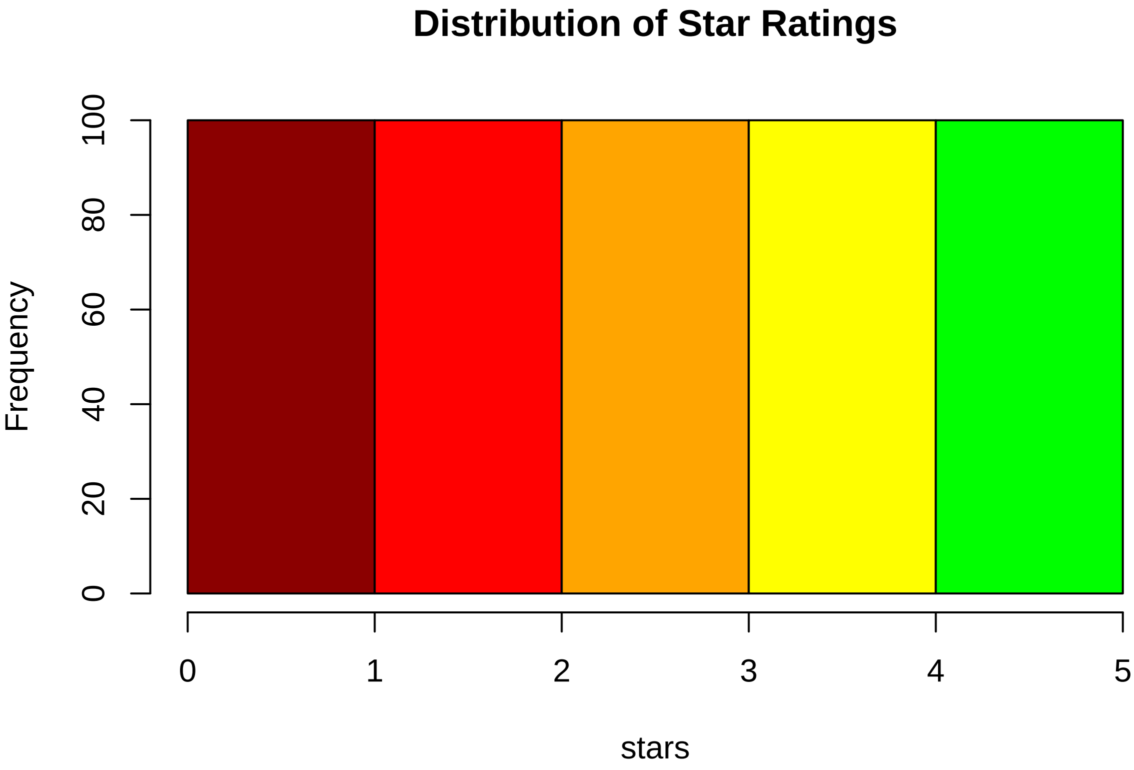
```
max(reviews_scraped$star_rating_num)
```

```
## [1] 5
```

The **worst rating** is 3 stars, the **highest one** is 5 stars and the **average rating** amounts to 4.57 stars. The star ratings indicate a very positive performance.

Another visualization of the **distribution of star ratings** might be helpful to see how the product has performed in general. Thus, here we present a histogram:

```
stars <- reviews_scraped$star_rating_num
cols <- c("dark red", "red", "orange", "yellow", "green")
hist(stars, main="Distribution of Star Ratings",xlab="stars", col=cols, breaks = 0:5)
```



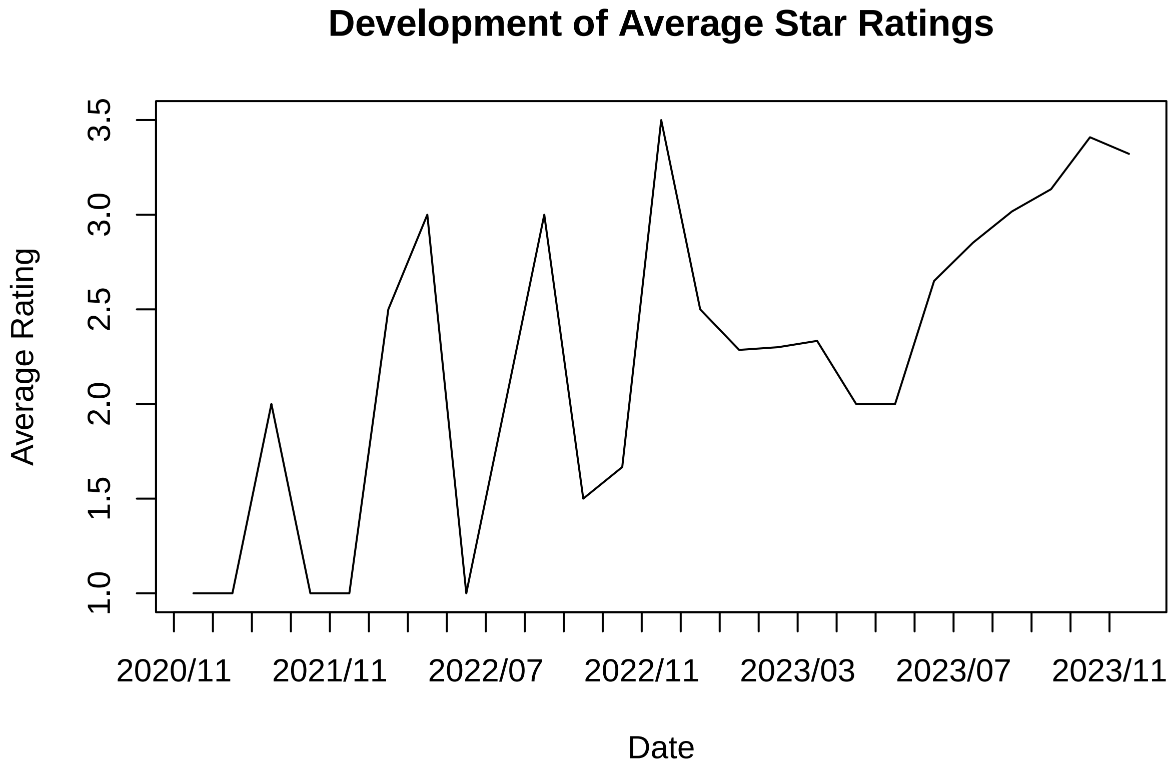
This shows how the star ratings are distributed in a bit more detail. We can see that the vast majority of reviews gave 4 or 5 stars to the product.

It might be also interesting to **reviews have changed over time**. For this we first summarize months.

```
dates <-strftime(reviews_scraped$formatted_dates, "%Y/%m")
```

Now we want to check whether the **average rating has improved or deteriorated** over time.

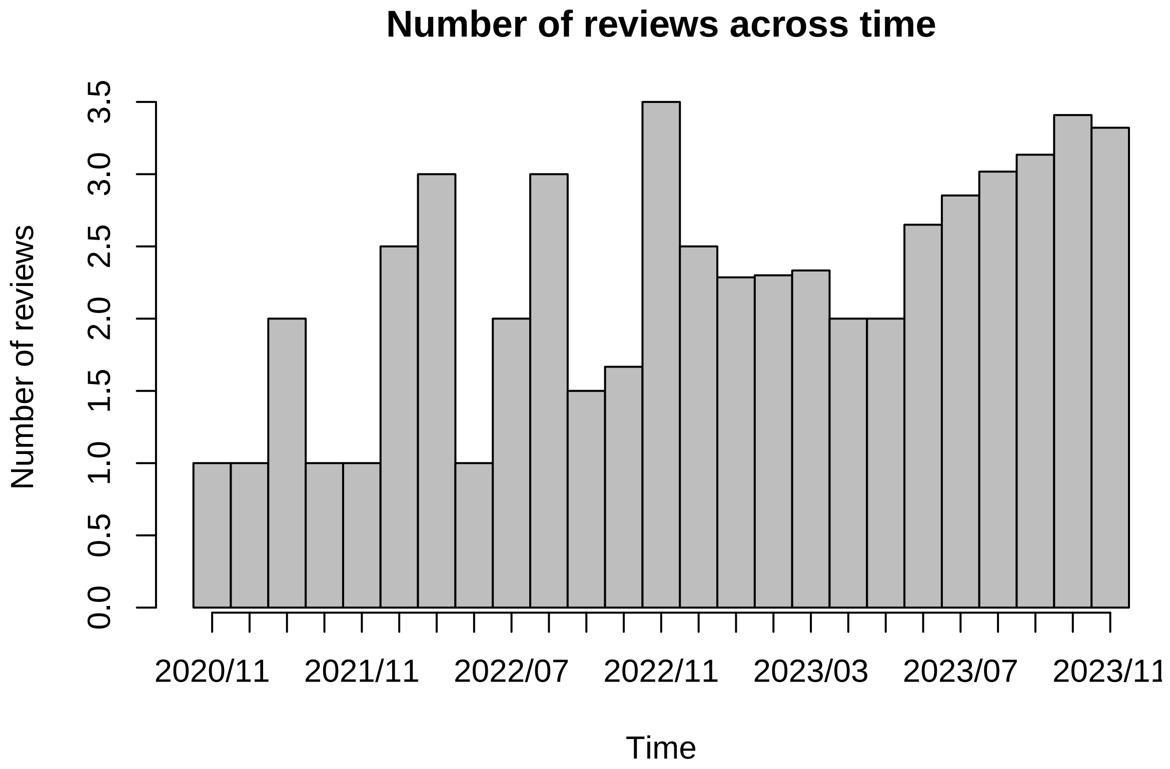
```
#plot rating distribution across time
plottingstars <- aggregate(reviews_scraped$star_rating_num ~ dates, FUN = mean)
plot(plottingstars[,2],type="l" ,xlab="Date",xaxt="n",ylab="Average Rating"
     ,main="Development of Average Star Ratings")
axis(side=1,at=1:nrow(plottingstars)-0.5,labels=plottingstars[1:nrow(plottingstars),1])
```



It seems as the average rating improves already after the first reviews and stays relatively constant throughout the time period where reviews were published.

It might also be interesting to take a look at **the number of reviews published across our time period** and see if and how this corresponds to the development of the star ratings.

```
# plot number of reviews distribution across time
plottingcount <- aggregate(reviews_scraped$star_rating_num ~ dates, FUN = length)
barplot(plottingstars[,2], main="Number of reviews across time",xlab="Time",xaxt="n",
        ylab="Number of reviews", space=0)
axis(side=1,at=1:nrow(plottingcount)-0.5,labels=plottingcount[1:nrow(plottingcount),1])
```



It appears that in all time periods a somewhat equal amount of ratings has been published and never below 4.However there is an increasing tendency of the reviews.

When comparing the two plots we can see that overall the review-situation improved over time: Both more reviews and better reviews were published in November than for example in February or June. The dip in June might be interesting to analyze - especially to see if it remains so apparent when the dataset of reviews is larger.

Furthermore, we might want to take a look at **helpfulness of the reviews**. For this we start off with simply wanting to know how the reviews were distribtued in this aspect.

```
reviews_scraped$N_helpful[is.na(reviews_scraped$N_helpful)] <- 0
min(reviews_scraped$N_helpful)
```

```
## [1] 0
```

```
mean(reviews_scraped$N_helpful)
```

```
## [1] 2.088
```

```
median(reviews_scraped$N_helpful)
```

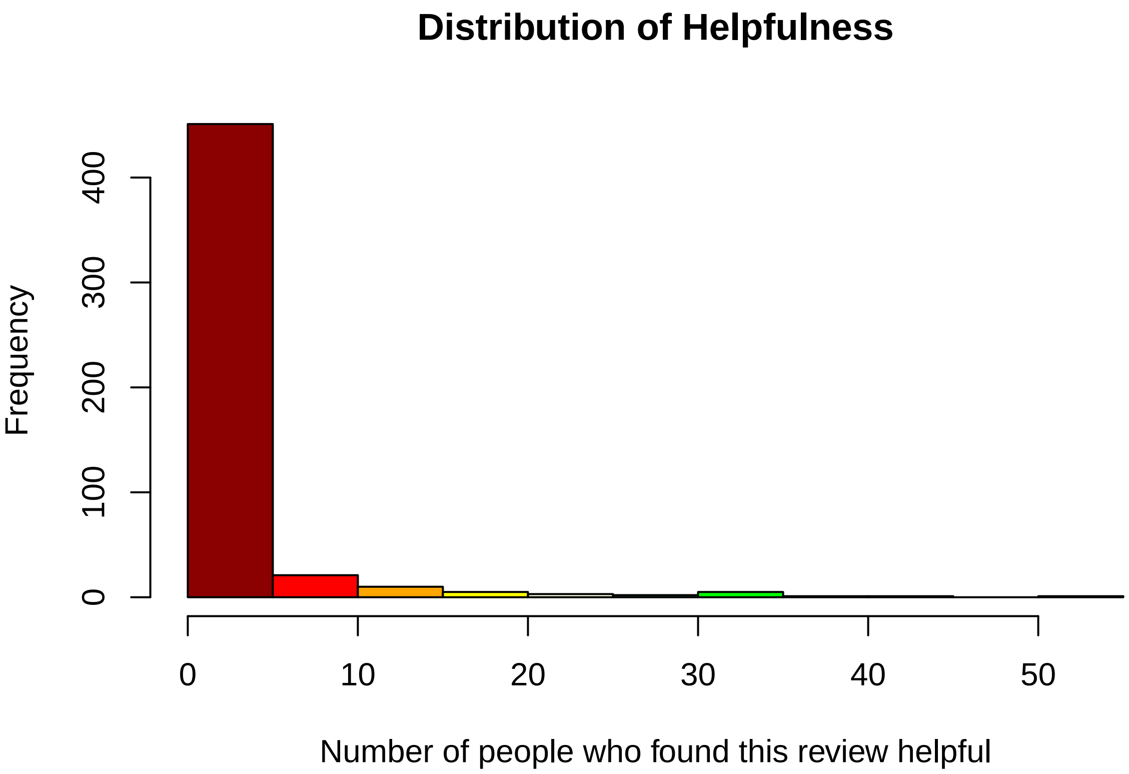
```
## [1] 0
```

```
max(reviews_scraped$N_helpful)
```

```
## [1] 51
```

We can see that the reviews were rated as helpful by between 0 and 50 people, with an average of 3.47 and a median of 0 as well. This indicates skewedness and strong outliers, so we want to take a closer look.

```
helpful <- reviews_scraped$N_helpful
cols <- c("dark red", "red", "orange", "yellow", "light yellow", "light green", "green", "dark
hist(helpful, main="Distribution of Helpfulness",xlab="Number of people who found this review
```



This supports our hypothesis that the vast majority of reviews are rated as non-helpful by very few to none people. Only very few people attained over 15 "helpfulness"-votes.

This comes down to the following general helpfulness of reviews:

```
dim(reviews_scraped[reviews_scraped$N_helpful>0,])[1]/dim(reviews_scraped)[1]
```

```
## [1] 0.354
```

Overall a third of the reviews were perceived as helpful, meaning that at least 1 customer rated them as such.

Text Analysis

Now that we have taken a look at the ratings and developments over time of our reviews, let us dive deeper into **what the reviews actually say**:

In order to do some text analysis, we must first do some pre-processing of the reviews.

Let us start by looking at **how long the review texts are**:

```
reviews_scraped$review_text_2 <- reviews_scraped$review_text
reviews_scraped$review_text_2 <-  gsub('[[:punct:] ]+',' ',reviews_scraped$review_text_2)
#split into substrings
reviews_scraped$review_text_2 <-strsplit(reviews_scraped$review_text_2, " ")
```

```r
#count the number of strings = number of words
reviews_scraped$review_length <- sapply(reviews_scraped$review_text_2, length)
print(reviews_scraped$review_length)
```

```
##   [1] 370 301 222 117  95 136  69 153  58  48  71  30  39  22  39  51  22  41
##  [19]  40  35  49  23  20  50  53  78  19 273   5  14  13  67  48  24  18  48
##  [37]   2   1   0   8   1  12   7   7   7   6   7  21  16  10   3   3   4   3
##  [55]  14  65  14  59  50  11  17  11   4  23  84   2  16  52  25  34   3  10
##  [73]  93  63  59  10  17  29  12  11 198   7   4  11  16  17   6   5  16   3
##  [91]   3  17  11   2   2  42  22  13   1  27 422 197  78 192 339  39  57  76
## [109]  67  15  10 185  36  17  17  15   8  11  67  42  25  74  22  64  60  31
## [127]  12  26   2  12  84  63  20  16  10  45  53 126  55   3  29  11  15  52
## [145]  41 100  15  50  54   5  11  66 176  12  80  21  77 232  54  85  50  13
## [163]  21  22   8   5  12  15  56 100  28   9  27 131  31  62  16   4 166   2
## [181] 211  10  13  17  69  44  65  39  25  48  24  60   1  32   6  25  37   4
## [199]   5 103 220 390 169  50  37  88  32  79  27  26  24  76  23  24  66  26
## [217]  22  19  65  16   8  36  31  27  30  24  22 293  27  21  61 125  10  47
## [235]  83  12 102  37   5  54  92   4  61  47   4  84 248  26 181  45  17  95
## [253]  42  52 108  34  21  51   7  42 112 105  10  64  31  43  56  67  48  15
## [271]  40  42 175  25  12  31 145  86  65 140 132 116  65  32 112  67   1  34
## [289] 200  25 187 115  23  11  31  11  30  17  22  59 138 244  92 110  69  60
## [307]  55 285  53  52  43  64  64 209  56  33 136  30  48  29  44 113  19  55
## [325]  21  14  52  57 105  25  47  41  42 150 133  39  39  69  40  34  14  14
## [343]  80  80  53  91  52  77  66  65 184  21  82  42  22  40  82  29  17 174
## [361]  20  78  25  44  10  43  22   8  29  10  21  49  82  62  59  20  83  16
## [379] 137  49 168  59   7  27  21 134  65  19  50   9  86  18  72 183  77  37
## [397]  73  44  29  61 507 325 347 256 385 313 267 205 189 272 374 144 244 288
## [415] 133 115 164 108 202 227 126 323 183 266  82  84 201 159  80  76  94 171
## [433]  90  70  66  87  57  62 274 160 307  84 176  86  52  47 163  50  51  60
## [451]  47  45  43  42  41  39  45 168  42  38 212  35  35  72 192  32  66 204
## [469] 167 121  66  66  64  36  63  64  62  81  58 103  57  33  53 280  48  28
## [487]  31 174  96 210 111  46  48 105  29  27  28  86  96  27
```

Given this, we can for example now check the average length of reviews:

```r
mean(reviews_scraped$review_length)
```

```
## [1] 70.392
```

We find out that the average review text is around 65 words long.

What might be more interesting is to see if there is some **correlation between the length of a reviews and its rating**:

```r
correlation <- cor(reviews_scraped$review_length, reviews_scraped$star_rating_num)
correlation
```

```
## [1] -0.3355965
```

The correlation is -0.271080. What this reveals is that long reviews tend to be more negative or that customers who have a negative opinion about a product tend to share more information to explain or complain.

Let's see if there is some **correlation between the length of a review and its helpfulness**:

```
correlation <- cor(reviews_scraped$review_length, reviews_scraped$N_helpful)
correlation
```

```
## [1] 0.6302964
```

The correlation is at 0.7532371. This leads to the conclusion that longer reviews tend to be rated as more helpful as they for example share more information and details that can be useful for other potential customers.

Now let us dive even deeper into text analysis than merely the word count. Let's have a look at **what people actually write in their reviews**.

```
# we start by loading in our text data as a "corpus"
TextDoc <-  Corpus(VectorSource((reviews_scraped$review_text_2)))

#Replacing "/", "@" and "|" with space
toSpace <- content_transformer(function (x , pattern ) gsub(pattern, " ", x))
TextDoc <- tm_map(TextDoc, toSpace, "/")
```

```
## Warning in tm_map.SimpleCorpus(TextDoc, toSpace, "/"): transformation drops
## documents
```

```
TextDoc <- tm_map(TextDoc, toSpace, "@")
```

```
## Warning in tm_map.SimpleCorpus(TextDoc, toSpace, "@"): transformation drops
## documents
```

```
TextDoc <- tm_map(TextDoc, toSpace, "\\|")
```

```
## Warning in tm_map.SimpleCorpus(TextDoc, toSpace, "\\|"): transformation drops
## documents
```

```
# Convert the text to lower case
TextDoc <- tm_map(TextDoc, content_transformer(tolower))
```

```
## Warning in tm_map.SimpleCorpus(TextDoc, content_transformer(tolower)):
## transformation drops documents
```

```
# Remove numbers
TextDoc <- tm_map(TextDoc, removeNumbers)
```

```
## Warning in tm_map.SimpleCorpus(TextDoc, removeNumbers): transformation drops
## documents
```

```
# Eliminate extra white spaces
TextDoc <- tm_map(TextDoc, stripWhitespace)
```

```
## Warning in tm_map.SimpleCorpus(TextDoc, stripWhitespace): transformation drops
## documents
```

```
# Remove English common stopwords
TextDoc <- tm_map(TextDoc, removeWords, stopwords("english"))
```

```
## Warning in tm_map.SimpleCorpus(TextDoc, removeWords, stopwords("english")):
## transformation drops documents
```

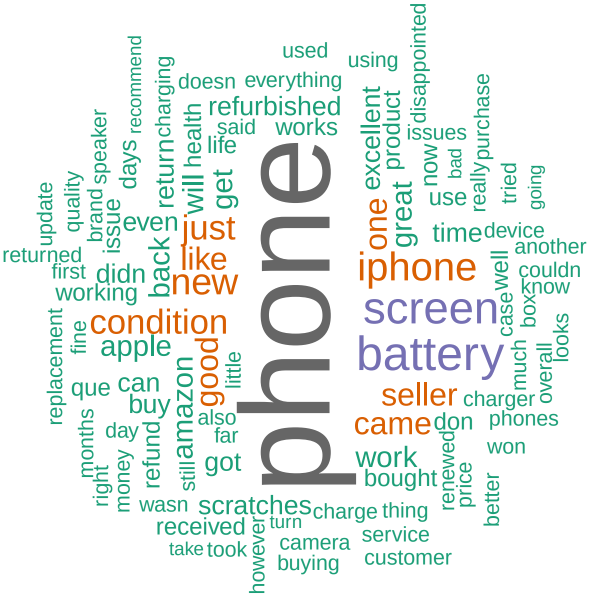
```
# Text stemming - which reduces words to their root form
TextDoc <- tm_map(TextDoc, stemDocument)
```

```
## Warning in tm_map.SimpleCorpus(TextDoc, stemDocument): transformation drops
## documents
```

```
# Remove all punctuation
TextDoc <- tm_map(TextDoc, removePunctuation)
```

```
## Warning in tm_map.SimpleCorpus(TextDoc, removePunctuation): transformation
## drops documents
```

```
# we continue to now build a term-document matrix
TextDoc_tdm <- TermDocumentMatrix(TextDoc)
tdm_m <- as.matrix(TextDoc_tdm)
# we sort the terms by decreasing frequency
tdm_v <- sort(rowSums(tdm_m),decreasing=TRUE)
tdm_d <- data.frame(word = names(tdm_v),freq=tdm_v)

# with these steps accomplished we can now build a word cloud
set.seed(1234)
wordcloud(words = tdm_d$word, freq = tdm_d$freq, min.freq = 5,
          max.words=100, random.order=FALSE, rot.per=0.40,
          colors=brewer.pal(8, "Dark2"))
```



Words such as “iphone”, “phone” or “apple” do not surprise us as being frequently used since they refer to product name and brand. If we want to make room for new revelations, we can remove them.

Therefore we will make further alterations to remove chosen words such as the product name itself - this wors via custom stop words:

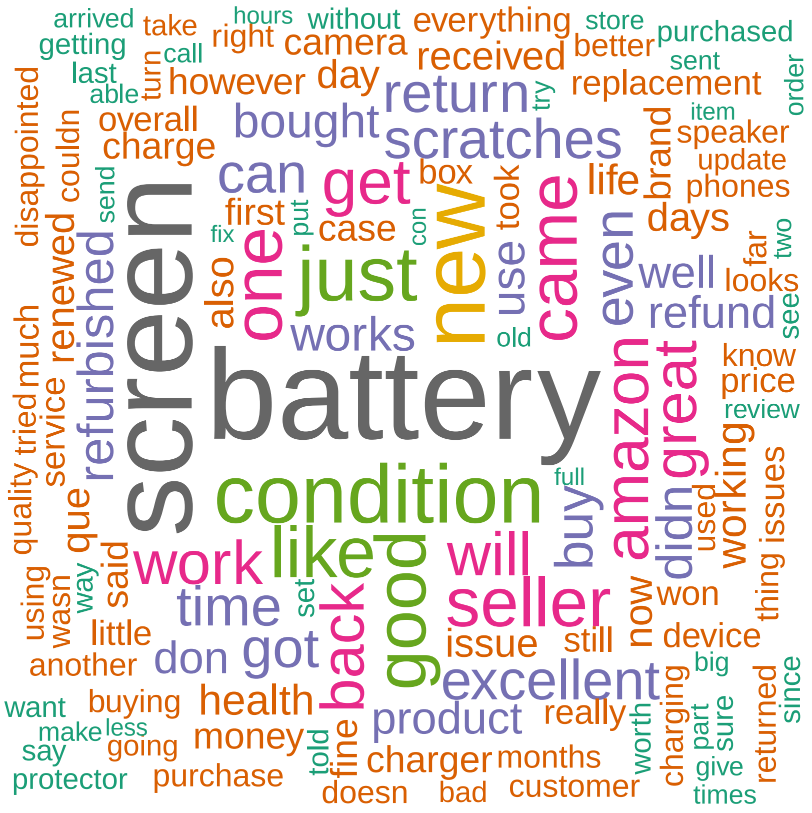
```
# specify your custom stopwords as a character vector
TextDoc <- tm_map(TextDoc, removeWords, c("iphone","phone","apple"))
```

```
## Warning in tm_map.SimpleCorpus(TextDoc, removeWords, c("iphone", "phone", :
## transformation drops documents
```

```
# Build a term-document matrix
TextDoc_tdm <- TermDocumentMatrix(TextDoc)
tdm_m <- as.matrix(TextDoc_tdm)
# Sort by decreasing value of frequency
tdm_v <- sort(rowSums(tdm_m),decreasing=TRUE)
tdm_d <- data.frame(word = names(tdm_v),freq=tdm_v)

#generate word cloud
set.seed(1234)
wordcloud(words = tdm_d$word, freq = tdm_d$freq, min.freq = 5,
          max.words=200, random.order=FALSE, rot.per=0.40,
          colors=brewer.pal(8, "Dark2"))
```

```
## Warning in wordcloud(words = tdm_d$word, freq = tdm_d$freq, min.freq = 5, :
## recommend could not be fit on page. It will not be plotted.

## Warning in wordcloud(words = tdm_d$word, freq = tdm_d$freq, min.freq = 5, :
## thought could not be fit on page. It will not be plotted.

## Warning in wordcloud(words = tdm_d$word, freq = tdm_d$freq, min.freq = 5, :
## happy could not be fit on page. It will not be plotted.

## Warning in wordcloud(words = tdm_d$word, freq = tdm_d$freq, min.freq = 5, :
## ordered could not be fit on page. It will not be plotted.

## Warning in wordcloud(words = tdm_d$word, freq = tdm_d$freq, min.freq = 5, :
## started could not be fit on page. It will not be plotted.

## Warning in wordcloud(words = tdm_d$word, freq = tdm_d$freq, min.freq = 5, :
## cphone could not be fit on page. It will not be plotted.

## Warning in wordcloud(words = tdm_d$word, freq = tdm_d$freq, min.freq = 5, :
## weeks could not be fit on page. It will not be plotted.

## Warning in wordcloud(words = tdm_d$word, freq = tdm_d$freq, min.freq = 5, :
## long could not be fit on page. It will not be plotted.

## Warning in wordcloud(words = tdm_d$word, freq = tdm_d$freq, min.freq = 5, :
## problem could not be fit on page. It will not be plotted.

## Warning in wordcloud(words = tdm_d$word, freq = tdm_d$freq, min.freq = 5, :
## experience could not be fit on page. It will not be plotted.

## Warning in wordcloud(words = tdm_d$word, freq = tdm_d$freq, min.freq = 5, :
```

```
## worked could not be fit on page. It will not be plotted.

## Warning in wordcloud(words = tdm_d$word, freq = tdm_d$freq, min.freq = 5, :
## hear could not be fit on page. It will not be plotted.

## Warning in wordcloud(words = tdm_d$word, freq = tdm_d$freq, min.freq = 5, :
## unlocked could not be fit on page. It will not be plotted.

## Warning in wordcloud(words = tdm_d$word, freq = tdm_d$freq, min.freq = 5, :
## something could not be fit on page. It will not be plotted.

## Warning in wordcloud(words = tdm_d$word, freq = tdm_d$freq, min.freq = 5, :
## touch could not be fit on page. It will not be plotted.

## Warning in wordcloud(words = tdm_d$word, freq = tdm_d$freq, min.freq = 5, :
## shipping could not be fit on page. It will not be plotted.

## Warning in wordcloud(words = tdm_d$word, freq = tdm_d$freq, min.freq = 5, :
## replaced could not be fit on page. It will not be plotted.

## Warning in wordcloud(words = tdm_d$word, freq = tdm_d$freq, min.freq = 5, :
## anything could not be fit on page. It will not be plotted.

## Warning in wordcloud(words = tdm_d$word, freq = tdm_d$freq, min.freq = 5, :
## company could not be fit on page. It will not be plotted.

## Warning in wordcloud(words = tdm_d$word, freq = tdm_d$freq, min.freq = 5, :
## definitely could not be fit on page. It will not be plotted.

## Warning in wordcloud(words = tdm_d$word, freq = tdm_d$freq, min.freq = 5, :
## capacity could not be fit on page. It will not be plotted.

## Warning in wordcloud(words = tdm_d$word, freq = tdm_d$freq, min.freq = 5, :
## different could not be fit on page. It will not be plotted.

## Warning in wordcloud(words = tdm_d$word, freq = tdm_d$freq, min.freq = 5, :
## never could not be fit on page. It will not be plotted.

## Warning in wordcloud(words = tdm_d$word, freq = tdm_d$freq, min.freq = 5, :
## though could not be fit on page. It will not be plotted.

## Warning in wordcloud(words = tdm_d$word, freq = tdm_d$freq, min.freq = 5, :
## expected could not be fit on page. It will not be plotted.

## Warning in wordcloud(words = tdm_d$word, freq = tdm_d$freq, min.freq = 5, :
## went could not be fit on page. It will not be plotted.

## Warning in wordcloud(words = tdm_d$word, freq = tdm_d$freq, min.freq = 5, :
## need could not be fit on page. It will not be plotted.

## Warning in wordcloud(words = tdm_d$word, freq = tdm_d$freq, min.freq = 5, :
## cost could not be fit on page. It will not be plotted.

## Warning in wordcloud(words = tdm_d$word, freq = tdm_d$freq, min.freq = 5, :
## minutes could not be fit on page. It will not be plotted.
## Warning in wordcloud(words = tdm_d$word, freq = tdm_d$freq, min.freq = 5, :
## perfectly could not be fit on page. It will not be plotted.

## Warning in wordcloud(words = tdm_d$word, freq = tdm_d$freq, min.freq = 5, :
## looked could not be fit on page. It will not be plotted.

## Warning in wordcloud(words = tdm_d$word, freq = tdm_d$freq, min.freq = 5, :
## black could not be fit on page. It will not be plotted.

## Warning in wordcloud(words = tdm_d$word, freq = tdm_d$freq, min.freq = 5, :
## replace could not be fit on page. It will not be plotted.

## Warning in wordcloud(words = tdm_d$word, freq = tdm_d$freq, min.freq = 5, :
## calls could not be fit on page. It will not be plotted.

## Warning in wordcloud(words = tdm_d$word, freq = tdm_d$freq, min.freq = 5, :
## problems could not be fit on page. It will not be plotted.

## Warning in wordcloud(words = tdm_d$word, freq = tdm_d$freq, min.freq = 5, :
## think could not be fit on page. It will not be plotted.

## Warning in wordcloud(words = tdm_d$word, freq = tdm_d$freq, min.freq = 5, :
## perfect could not be fit on page. It will not be plotted.

## Warning in wordcloud(words = tdm_d$word, freq = tdm_d$freq, min.freq = 5, :
## stopped could not be fit on page. It will not be plotted.

## Warning in wordcloud(words = tdm_d$word, freq = tdm_d$freq, min.freq = 5, :
## left could not be fit on page. It will not be plotted.

## Warning in wordcloud(words = tdm_d$word, freq = tdm_d$freq, min.freq = 5, :
## month could not be fit on page. It will not be plotted.

## Warning in wordcloud(words = tdm_d$word, freq = tdm_d$freq, min.freq = 5, :
## original could not be fit on page. It will not be plotted.

## Warning in wordcloud(words = tdm_d$word, freq = tdm_d$freq, min.freq = 5, :
## hope could not be fit on page. It will not be plotted.

## Warning in wordcloud(words = tdm_d$word, freq = tdm_d$freq, min.freq = 5, :
## repair could not be fit on page. It will not be plotted.

## Warning in wordcloud(words = tdm_d$word, freq = tdm_d$freq, min.freq = 5, :
## completely could not be fit on page. It will not be plotted.

## Warning in wordcloud(words = tdm_d$word, freq = tdm_d$freq, min.freq = 5, :
## reviews could not be fit on page. It will not be plotted.

## Warning in wordcloud(words = tdm_d$word, freq = tdm_d$freq, min.freq = 5, :
## someone could not be fit on page. It will not be plotted.

## Warning in wordcloud(words = tdm_d$word, freq = tdm_d$freq, min.freq = 5, :
## power could not be fit on page. It will not be plotted.

## Warning in wordcloud(words = tdm_d$word, freq = tdm_d$freq, min.freq = 5, :
## already could not be fit on page. It will not be plotted.
## Warning in wordcloud(words = tdm_d$word, freq = tdm_d$freq, min.freq = 5, : pay
## could not be fit on page. It will not be plotted.

## Warning in wordcloud(words = tdm_d$word, freq = tdm_d$freq, min.freq = 5, :
## paid could not be fit on page. It will not be plotted.

## Warning in wordcloud(words = tdm_d$word, freq = tdm_d$freq, min.freq = 5, :
## apps could not be fit on page. It will not be plotted.

## Warning in wordcloud(words = tdm_d$word, freq = tdm_d$freq, min.freq = 5, :
## small could not be fit on page. It will not be plotted.

## Warning in wordcloud(words = tdm_d$word, freq = tdm_d$freq, min.freq = 5, :
## scratch could not be fit on page. It will not be plotted.

## Warning in wordcloud(words = tdm_d$word, freq = tdm_d$freq, min.freq = 5, :
## best could not be fit on page. It will not be plotted.

## Warning in wordcloud(words = tdm_d$word, freq = tdm_d$freq, min.freq = 5, :
## second could not be fit on page. It will not be plotted.

## Warning in wordcloud(words = tdm_d$word, freq = tdm_d$freq, min.freq = 5, :
## pero could not be fit on page. It will not be plotted.

## Warning in wordcloud(words = tdm_d$word, freq = tdm_d$freq, min.freq = 5, :
## settings could not be fit on page. It will not be plotted.

## Warning in wordcloud(words = tdm_d$word, freq = tdm_d$freq, min.freq = 5, :
## button could not be fit on page. It will not be plotted.

## Warning in wordcloud(words = tdm_d$word, freq = tdm_d$freq, min.freq = 5, :
## seems could not be fit on page. It will not be plotted.

## Warning in wordcloud(words = tdm_d$word, freq = tdm_d$freq, min.freq = 5, :
## week could not be fit on page. It will not be plotted.

## Warning in wordcloud(words = tdm_d$word, freq = tdm_d$freq, min.freq = 5, :
## noticeable could not be fit on page. It will not be plotted.

## Warning in wordcloud(words = tdm_d$word, freq = tdm_d$freq, min.freq = 5, :
## option could not be fit on page. It will not be plotted.

## Warning in wordcloud(words = tdm_d$word, freq = tdm_d$freq, min.freq = 5, :
## satisfied could not be fit on page. It will not be plotted.

## Warning in wordcloud(words = tdm_d$word, freq = tdm_d$freq, min.freq = 5, :
## fast could not be fit on page. It will not be plotted.

## Warning in wordcloud(words = tdm_d$word, freq = tdm_d$freq, min.freq = 5, :
## otherwise could not be fit on page. It will not be plotted.

## Warning in wordcloud(words = tdm_d$word, freq = tdm_d$freq, min.freq = 5, :
## loaded could not be fit on page. It will not be plotted.

## Warning in wordcloud(words = tdm_d$word, freq = tdm_d$freq, min.freq = 5, :
## message could not be fit on page. It will not be plotted.
```



This reveals a much more interesting picture. We chose that all terms that are mentioned at least 5 times in the 100 reviews are shown in the word cloud. The by far biggest and thus most frequently used significant term in reviews is “battery”. Additionally screen and scratches as well as condition might be interesting ones to dive into.

Let us now do the same for the titles of the reviews and see if there are differences:

```
# we start by loading in our text data as a "corpus"
TextDoc <-  Corpus(VectorSource((reviews_scraped$review_title)))

#Replacing "/", "@" and "|" with space
toSpace <- content_transformer(function (x , pattern ) gsub(pattern, " ", x))
TextDoc <- tm_map(TextDoc, toSpace, "/")
```

```
## Warning in tm_map.SimpleCorpus(TextDoc, toSpace, "/"): transformation drops
## documents
```

```
TextDoc <- tm_map(TextDoc, toSpace, "@")
```

```
## Warning in tm_map.SimpleCorpus(TextDoc, toSpace, "@"): transformation drops
## documents
```

```
TextDoc <- tm_map(TextDoc, toSpace, "\\|")
```

```
## Warning in tm_map.SimpleCorpus(TextDoc, toSpace, "\\|"): transformation drops
## documents
```

```r
# Convert the text to lower case
TextDoc <- tm_map(TextDoc, content_transformer(tolower))
```

```
## Warning in tm_map.SimpleCorpus(TextDoc, content_transformer(tolower)):
## transformation drops documents
```

```r
# Remove numbers
TextDoc <- tm_map(TextDoc, removeNumbers)
```

```
## Warning in tm_map.SimpleCorpus(TextDoc, removeNumbers): transformation drops
## documents
```

```r
# Eliminate extra white spaces
TextDoc <- tm_map(TextDoc, stripWhitespace)
```

```
## Warning in tm_map.SimpleCorpus(TextDoc, stripWhitespace): transformation drops
## documents
```

```r
# Remove german common stopwords
TextDoc <- tm_map(TextDoc, removeWords, stopwords("english"))
```

```
## Warning in tm_map.SimpleCorpus(TextDoc, removeWords, stopwords("english")):
## transformation drops documents
```

```r
# Text stemming - which reduces words to their root form
TextDoc <- tm_map(TextDoc, stemDocument)
```

```
## Warning in tm_map.SimpleCorpus(TextDoc, stemDocument): transformation drops
## documents
```

```r
# Remove all punctuation
TextDoc <- tm_map(TextDoc, removePunctuation)
```

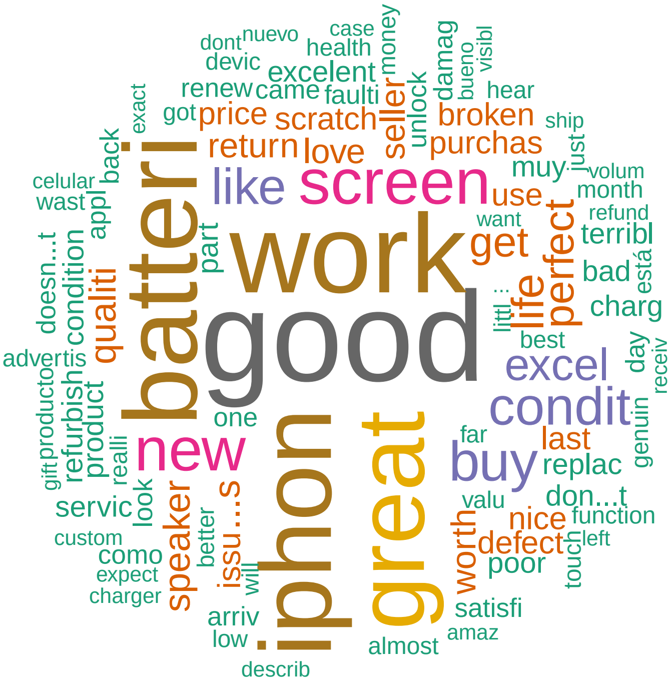
```
## Warning in tm_map.SimpleCorpus(TextDoc, removePunctuation): transformation
## drops documents
```

```r
# specify your custom stopwords as a character vector
TextDoc <- tm_map(TextDoc, removeWords, c("iphone","phone","apple"))
```

```
## Warning in tm_map.SimpleCorpus(TextDoc, removeWords, c("iphone", "phone", :
## transformation drops documents
```

```r
# Build a term-document matrix
TextDoc_tdm <- TermDocumentMatrix(TextDoc)
tdm_m <- as.matrix(TextDoc_tdm)
# Sort by decreasing value of frequency
tdm_v <- sort(rowSums(tdm_m),decreasing=TRUE)
tdm_d <- data.frame(word = names(tdm_v),freq=tdm_v)

# with these steps accomplished we can now build a word cloud
set.seed(1234)
wordcloud(words = tdm_d$word, freq = tdm_d$freq, min.freq = 2,
          max.words=100, random.order=FALSE, rot.per=0.40,
          colors=brewer.pal(8, "Dark2"))
```

```
## Warning in strwidth(words[i], cex = size[i], ...): conversion failure on '’s'
## in 'mbcsToSbcs': dot substituted for <e2>

## Warning in strwidth(words[i], cex = size[i], ...): conversion failure on '’s'
## in 'mbcsToSbcs': dot substituted for <80>

## Warning in strwidth(words[i], cex = size[i], ...): conversion failure on '’s'
## in 'mbcsToSbcs': dot substituted for <99>

## Warning in text.default(x1, y1, words[i], cex = size[i], offset = 0, srt =
## rotWord * : conversion failure on '’s' in 'mbcsToSbcs': dot substituted for
## <e2>

## Warning in text.default(x1, y1, words[i], cex = size[i], offset = 0, srt =
## rotWord * : conversion failure on '’s' in 'mbcsToSbcs': dot substituted for
## <80>

## Warning in text.default(x1, y1, words[i], cex = size[i], offset = 0, srt =
## rotWord * : conversion failure on '’s' in 'mbcsToSbcs': dot substituted for
## <99>

## Warning in text.default(x1, y1, words[i], cex = size[i], offset = 0, srt =
## rotWord * : font metrics unknown for Unicode character U+2019

## Warning in strwidth(words[i], cex = size[i], ...): conversion failure on
## 'don’t' in 'mbcsToSbcs': dot substituted for <e2>

## Warning in strwidth(words[i], cex = size[i], ...): conversion failure on
## 'don’t' in 'mbcsToSbcs': dot substituted for <80>

## Warning in strwidth(words[i], cex = size[i], ...): conversion failure on
## 'don’t' in 'mbcsToSbcs': dot substituted for <99>

## Warning in text.default(x1, y1, words[i], cex = size[i], offset = 0, srt =
## rotWord * : conversion failure on 'don’t' in 'mbcsToSbcs': dot substituted for
## <e2>

## Warning in text.default(x1, y1, words[i], cex = size[i], offset = 0, srt =
## rotWord * : conversion failure on 'don’t' in 'mbcsToSbcs': dot substituted for
## <80>

## Warning in text.default(x1, y1, words[i], cex = size[i], offset = 0, srt =
## rotWord * : conversion failure on 'don’t' in 'mbcsToSbcs': dot substituted for
## <99>

## Warning in text.default(x1, y1, words[i], cex = size[i], offset = 0, srt =
## rotWord * : font metrics unknown for Unicode character U+2019

## Warning in strwidth(words[i], cex = size[i], ...): conversion failure on
## 'doesn’t' in 'mbcsToSbcs': dot substituted for <e2>

## Warning in strwidth(words[i], cex = size[i], ...): conversion failure on
## 'doesn’t' in 'mbcsToSbcs': dot substituted for <80>
## Warning in strwidth(words[i], cex = size[i], ...): conversion failure on
## 'doesn’t' in 'mbcsToSbcs': dot substituted for <99>

## Warning in text.default(x1, y1, words[i], cex = size[i], offset = 0, srt =
## rotWord * : conversion failure on 'doesn’t' in 'mbcsToSbcs': dot substituted
## for <e2>

## Warning in text.default(x1, y1, words[i], cex = size[i], offset = 0, srt =
## rotWord * : conversion failure on 'doesn’t' in 'mbcsToSbcs': dot substituted
## for <80>

## Warning in text.default(x1, y1, words[i], cex = size[i], offset = 0, srt =
## rotWord * : conversion failure on 'doesn’t' in 'mbcsToSbcs': dot substituted
## for <99>

## Warning in text.default(x1, y1, words[i], cex = size[i], offset = 0, srt =
## rotWord * : font metrics unknown for Unicode character U+2019

## Warning in strwidth(words[i], cex = size[i], ...): conversion failure on '...' in
## 'mbcsToSbcs': dot substituted for <e2>

## Warning in strwidth(words[i], cex = size[i], ...): conversion failure on '...' in
## 'mbcsToSbcs': dot substituted for <80>

## Warning in strwidth(words[i], cex = size[i], ...): conversion failure on '...' in
## 'mbcsToSbcs': dot substituted for <a6>

## Warning in text.default(x1, y1, words[i], cex = size[i], offset = 0, srt =
## rotWord * : conversion failure on '...' in 'mbcsToSbcs': dot substituted for <e2>

## Warning in text.default(x1, y1, words[i], cex = size[i], offset = 0, srt =
## rotWord * : conversion failure on '...' in 'mbcsToSbcs': dot substituted for <80>

## Warning in text.default(x1, y1, words[i], cex = size[i], offset = 0, srt =
## rotWord * : conversion failure on '...' in 'mbcsToSbcs': dot substituted for <a6>

## Warning in text.default(x1, y1, words[i], cex = size[i], offset = 0, srt =
## rotWord * : font metrics unknown for Unicode character U+2026
```



Since this word cloud is less informative, we will include more words (at lower frequency threshold) and see an overwhelming frequency of very positive terms (good, grate, perfect). This is a positive indicator.

Sentiment analysis

Before we dive into applying sentiment analysis on our dataset of reviews, we tried to recall the basics of the Sentiment package and its meaning in R. Essentially sentiment analysis works by differentiating between words depending on the sentiment that is attached to them. This is conducted via dictionaries consisting of lists of positive vs. negative words, or lists of more diverse emotions. Packages such as sentimentr in R work by scanning the text to see if words in the text match with any dictionary entries. The words are then assigned a value (>0 if the word is located on the positive list, <0 if it is on the negative one) - all values are added together and the average sentiment is determined. Important note: The packages to take the words before and after a term into account in order to assess its classification. This way valence shifters or negations can be included. Sentiment analysis can be applied in a number of fields and situations. Its use-cases range from social media monitoring, political campaigns, PR and market research.

We now want to start our first computations in the field of sentiment analysis to get a better picture about the general tonality and sentiment of the reviews we are examining. In this we will differentiate between title and text look at the correlation between them and their general behavior.

```r
sentences <- get_sentences(reviews_scraped$review_text)
reviews_scraped$sentiment_text=sentiment_by(sentences)

sentences_title <- get_sentences(reviews_scraped$review_title)
reviews_scraped$sentiment_title=sentiment_by(sentences_title)

summary(cbind(reviews_scraped$sentiment_text$ave_sentiment,reviews_scraped$sentiment_title$ave_
##        V1                  V2
##  Min.   :-0.54524   Min.   :-1.40729
##  1st Qu.:-0.04643   1st Qu.:-0.14434
##  Median : 0.05303   Median : 0.00000
##  Mean   : 0.11167   Mean   : 0.06231
##  3rd Qu.: 0.25000   3rd Qu.: 0.34820
##  Max.   : 1.42500   Max.   : 1.23744
```

```
cor.test(reviews_scraped$sentiment_text$ave_sentiment, reviews_scraped$sentiment_title$ave_sen
```

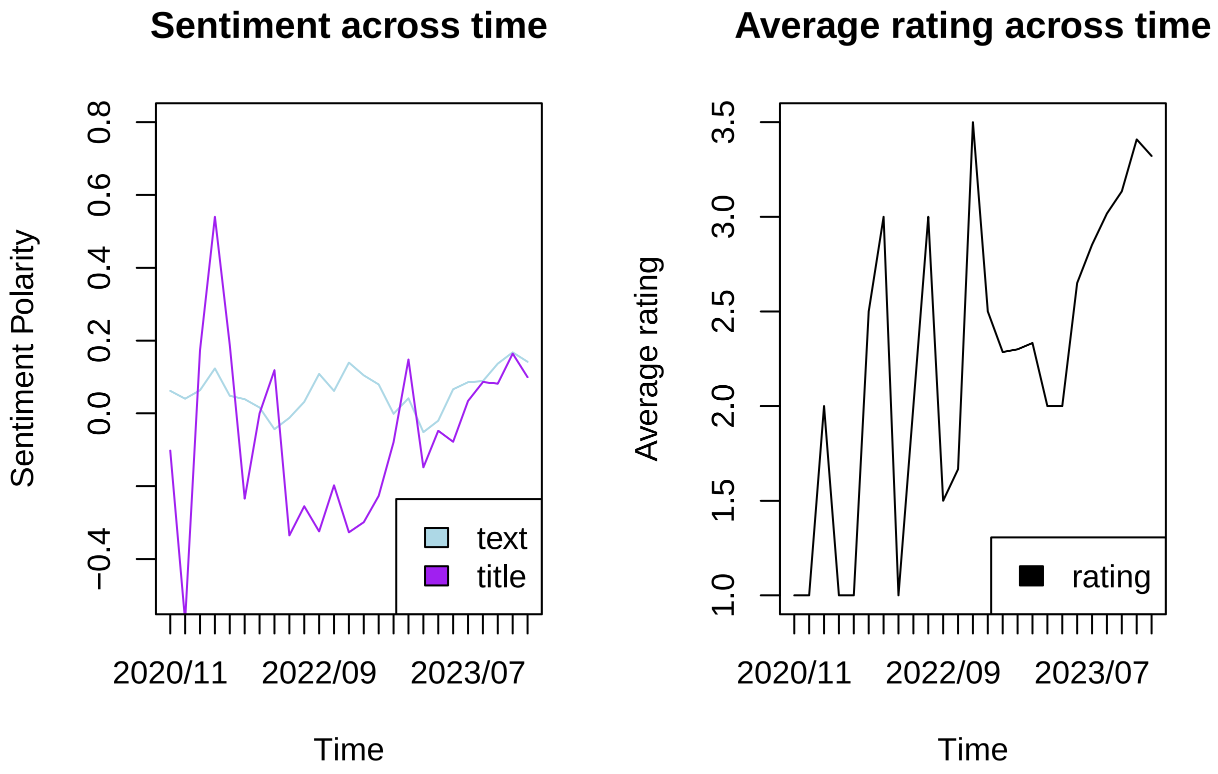
```
##
##  Pearson's product-moment correlation
##
## data:  reviews_scraped$sentiment_text$ave_sentiment and reviews_scraped$sentiment_title$ave_
## t = 11.062, df = 498, p-value < 2.2e-16
## alternative hypothesis: true correlation is not equal to 0
## 95 percent confidence interval:
##  0.3708724 0.5118749
## sample estimates:
##       cor
## 0.4441193
```

The computations reveal that in the review texts, sentiment ranges from -0.45 to 1.05 and averages at around 0.25. For the titles, the lowest sentiment is -0.6, the highest is 1.24 and the average is around 0.35. Both average sentiments are positive, which is good news for the product. Anything else would be highly unusual since the sentiment should correspond to star ratings and since there are no significant bad ratings, this would need further investigation. Simply comparing the numbers allows us to make the inference that in general the sentiment in titles is more extreme than that in texts. This should not surprise us - titles are meant to catch people's attentions and thus, similar as headlines in the news, use more aggressive wordings, stronger opinions and more catchy phrases. We also prove that the sentiment of title and review text are correlated since the p-value is extremely low.

Now, similar to the way we wanted to check if the rating behavior changed over time we want to take a look at the **development of sentiment over time**:

```
textsentiment_agg<-aggregate(reviews_scraped$sentiment_text$ave_sentiment ~ dates, FUN = mean)
titlesentiment_agg<-aggregate(reviews_scraped$sentiment_title$ave_sentiment ~ dates, FUN = mea
starratings_agg<-aggregate(reviews_scraped$star_rating_num ~ dates, FUN = mean)

par(mfrow=c(1,2))
plot(textsentiment_agg[,2],type="l" ,xlab="Time",xaxt="n",ylab="Sentiment Polarity", ylim=c(-0
     main="Sentiment across time",col="light blue")
lines(titlesentiment_agg[,2],type="l" ,xlab="Time",xaxt="n",ylab="Sentiment Polarity"
      ,main="Sentiment across time",col="purple")
axis(side=1,at=1:nrow(textsentiment_agg),labels=textsentiment_agg[1:nrow(textsentiment_agg),1]
legend("bottomright",c("text","title"),fill=c("light blue","purple"))

plot(starratings_agg[,2],type="l" ,xlab="Time",xaxt="n",ylab="Average rating"
     ,main="Average rating across time")
```

```
axis(side=1,at=1:nrow(starratings_agg),labels=starratings_agg[1:nrow(starratings_agg),1])
legend("bottomright","rating",fill="black")
```

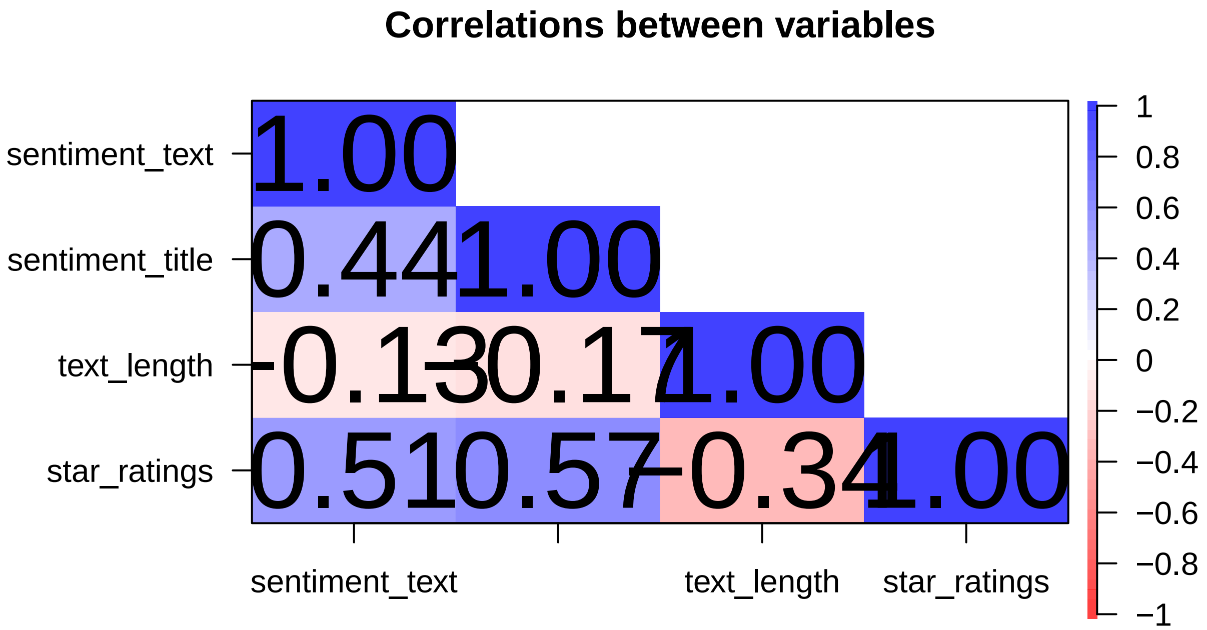


When comparing the plots of how sentiment (in both title and text) developed, we can see the same drop between title sentiment and ratings in June 2023. Additionally the stabilization of sentiment / slight improvement is mirrored in an increase in the average ratings.

We can take a look at how sentiment & other variables interact in more detail:

```
corrm <- cor(cbind(reviews_scraped$sentiment_text$ave_sentiment,reviews_scraped$sentiment_titl
colnames(corrm) <- c("sentiment_text","sentiment_title","text_length", "star_ratings")
rownames(corrm) <- c("sentiment_text","sentiment_title","text_length", "star_ratings")
corPlot(corrm ,numbers=T,diag=T,upper=F,  main="Correlations between variables", xact = "n")
```

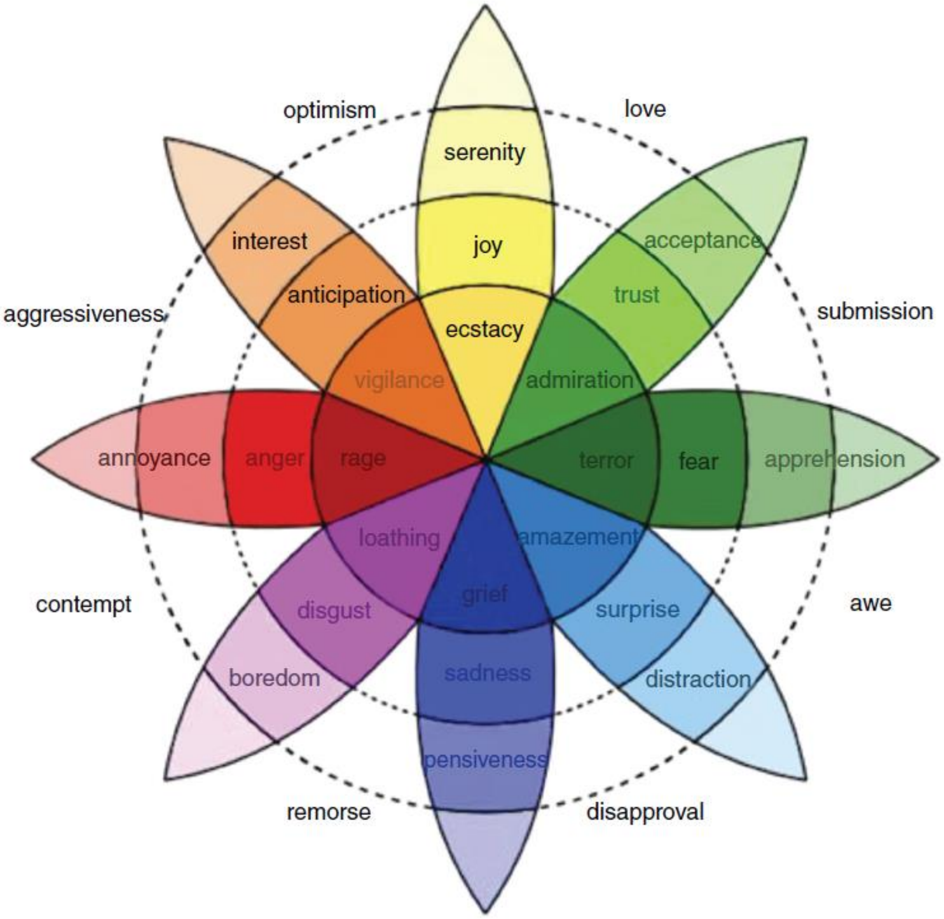
```
## Warning in axis(2, at = at2, labels = lab2, las = ylas, ...): "xact" is not a
## graphical parameter

## Warning in axis(xaxis, at = at1, labels = lab1, las = xlas, line = line, :
## "xact" is not a graphical parameter

## Warning in text.default(rx, ry, rv, cex = 1.5 * cex, ...): "xact" is not a
## graphical parameter

## Warning in axis(4, at = at2, labels = labels, las = 2, ...): "xact" is not a
## graphical parameter
```



From the correlation plot we can see a rather strong positive correlation between sentiment in title and sentiment in text - this we have already found out. Additionally we can see slight negative correlations between text length and sentiment. This further supports our findings that longer reviews will have a worse rating. Star ratings thus negatively correlate with text length, but are positively impacted by the sentiment score.

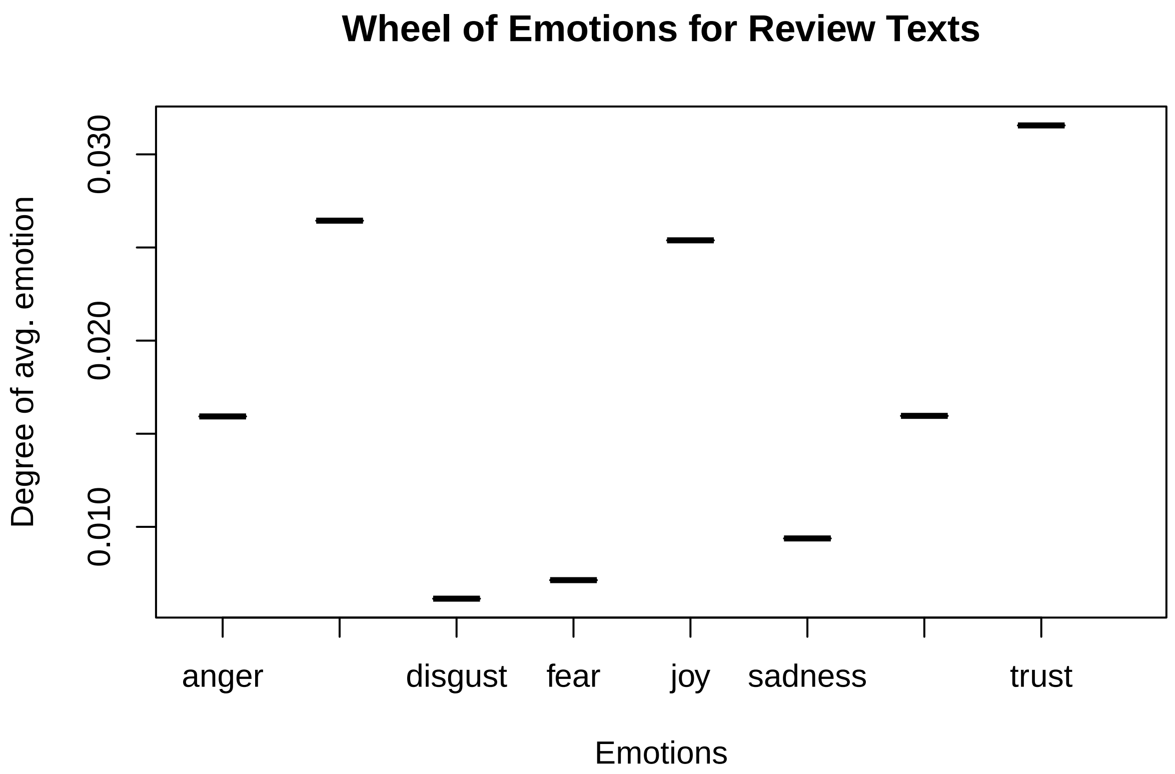
Emotion Analysis

In emotion analysis we get more detailed insights in the emotions that are expressed in reviews by differentiating not only between positively and negatively annotated terms, but a more developed spectrum of emotions. As a basis, Plutchik's wheel of emotion is used.

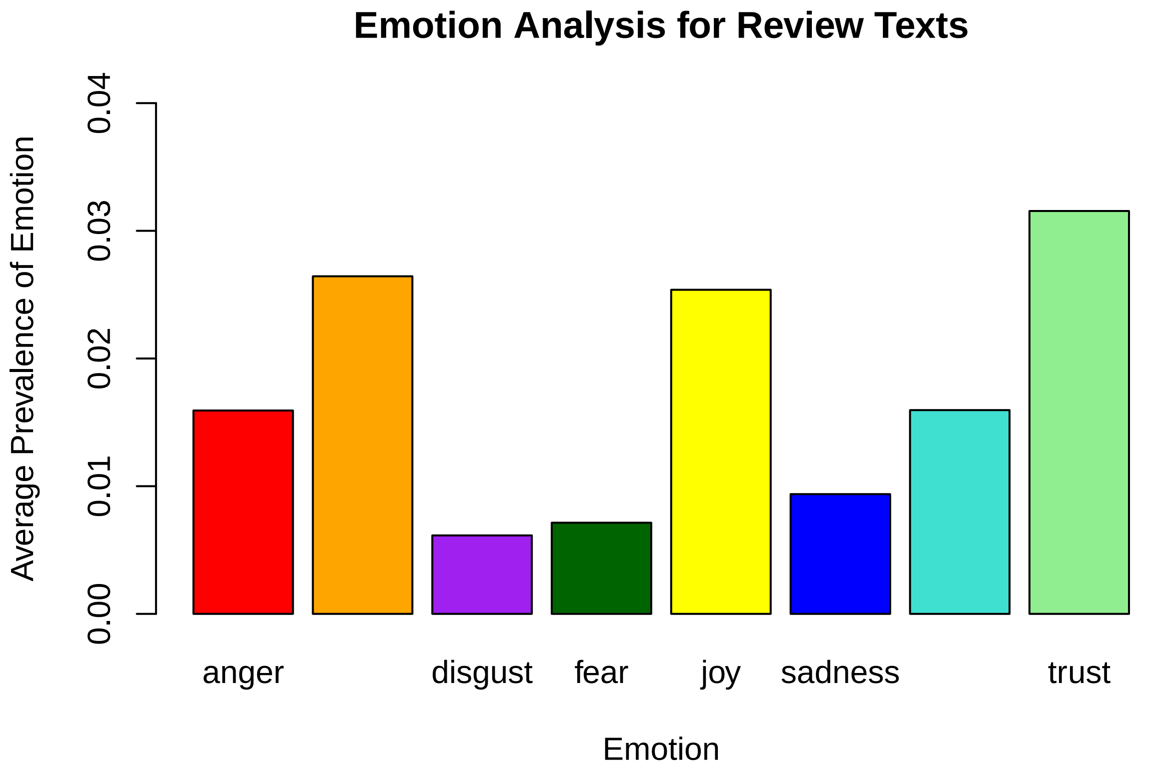


We now want to find out about the emotions in our reviews:

```
text_emotions <- emotion_by(sentences)
emotiontext_agg<-aggregate(text_emotions$ave_emotion ~ (text_emotions$emotion_type), FUN = mean
emotiontext_agg <- subset(emotiontext_agg,(1:dim(emotiontext_agg)[1])%%2==1) #look at non-nega
plot(emotiontext_agg,xaxt="n", ylab="Degree of avg. emotion",xlab="Emotions",main="Wheel of Emo
axis(side=1,at=row.names(emotiontext_agg),labels=emotiontext_agg[c(1:8),1])
```



```
colors = c("red", "orange", "purple", "dark green", "yellow", "blue", "turquoise", "light green
barplot(emotiontext_agg[, 2], names.arg = emotiontext_agg[, 1], col = colors, main="Emotion Ana
```



Essentially, this plot reveals information about how dominant certain emotions are in our review text. The higher the bar of a certain emotion, the higher its relevance. In the colored bar plot visualizations we can see which emotions correspond to which part of the wheel and can infer that the most dominant emotions in the review texts are anticipation (orange), joy (yellow) and trust (light green). However, there are significant degrees of anger as well.

Now let us take a closer look at the emotions separately:

```
#add emotions to data set
#reduce to non-negated emotions
text_emotions_nnegated <- text_emotions[-grep("negated",text_emotions$emotion_type),]

#get average emotions for each review
temp <- reshape(text_emotions_nnegated[,c(1,2,6)], idvar = "element_id", timevar = "emotion_typ
reviews_scraped <- cbind(reviews_scraped,temp)
emotions_list <- c("ave_emotion.anger", "ave_emotion.anticipation", "ave_emotion.disgust", "ave
summary(reviews_scraped[,emotions_list])
```

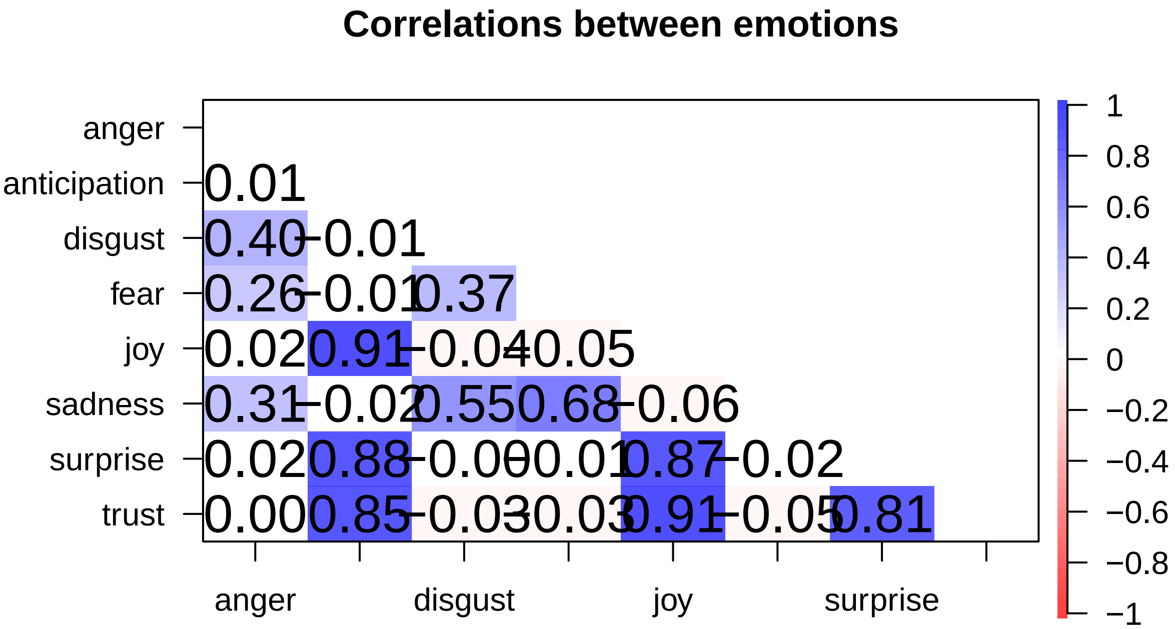
```
##  ave_emotion.anger  ave_emotion.anticipation ave_emotion.disgust
##  Min.   :0.000000   Min.   :0.00000          Min.   :0.000000
##  1st Qu.:0.000000   1st Qu.:0.00000          1st Qu.:0.000000
##  Median :0.001193   Median :0.01299          Median :0.000000
##  Mean   :0.015928   Mean   :0.02644          Mean   :0.006145
##  3rd Qu.:0.022990   3rd Qu.:0.03071          3rd Qu.:0.006501
##  Max.   :0.200000   Max.   :1.00000          Max.   :0.093750
##  ave_emotion.fear   ave_emotion.joy    ave_emotion.sadness ave_emotion.surprise
##  Min.   :0.000000   Min.   :0.000000   Min.   :0.000000    Min.   :0.00000
##  1st Qu.:0.000000   1st Qu.:0.000000   1st Qu.:0.000000    1st Qu.:0.00000
##  Median :0.000000   Median :0.009788   Median :0.000000    Median :0.00000
##  Mean   :0.007139   Mean   :0.025383   Mean   :0.009378    Mean   :0.01596
##  3rd Qu.:0.010583   3rd Qu.:0.029155   3rd Qu.:0.015152    3rd Qu.:0.01515
##  Max.   :0.111111   Max.   :1.000000   Max.   :0.084746    Max.   :1.00000
##  ave_emotion.trust
##  Min.   :0.00000
##  1st Qu.:0.00000
##  Median :0.01852
##  Mean   :0.03155
##  3rd Qu.:0.03704
##  Max.   :1.00000
```

Here we can see that the minimum of all emotions is 0 (every review features every emotion to at least some extent). Surprise reaches up to 0.125 and averages average at 0.013. Trust has a maximum of 0.2 and averages at 0.03. Anger has a maximum of 0.18 and averages at 0.115. Anticipation has a maximum of 0.2 and averages at 0.03. Disgust has a maximum of 0.07 and averages at 0.002. Feat has a maximum of 0.05 and averages at 0.003 and joy has a maximum of 0.2 and an average of 0.03. This again shows that trust, anticipation and joy have the highest means, thus are on average the prevalent emotions expressed in the reviews we are looking at.

Now we want to check if and how emotional categories are potentially related to one another:

```
corrE <- cor(reviews_scraped[,emotions_list])
colnames(corrE) <- c("anger","anticipation","disgust","fear","joy","sadness","surprise","trust"
rownames(corrE) <-c("anger","anticipation","disgust","fear","joy","sadness","surprise","trust")

corPlot(corrE ,numbers=T,diag=F,upper=F,  main="Correlations between emotions" ,xact="n" )
```

```
## Warning in axis(2, at = at2, labels = lab2, las = ylas, ...): "xact" is not a
## graphical parameter

## Warning in axis(xaxis, at = at1, labels = lab1, las = xlas, line = line, :
## "xact" is not a graphical parameter

## Warning in text.default(rx, ry, rv, cex = 1.5 * cex, ...): "xact" is not a
## graphical parameter

## Warning in axis(4, at = at2, labels = labels, las = 2, ...): "xact" is not a
## graphical parameter
```

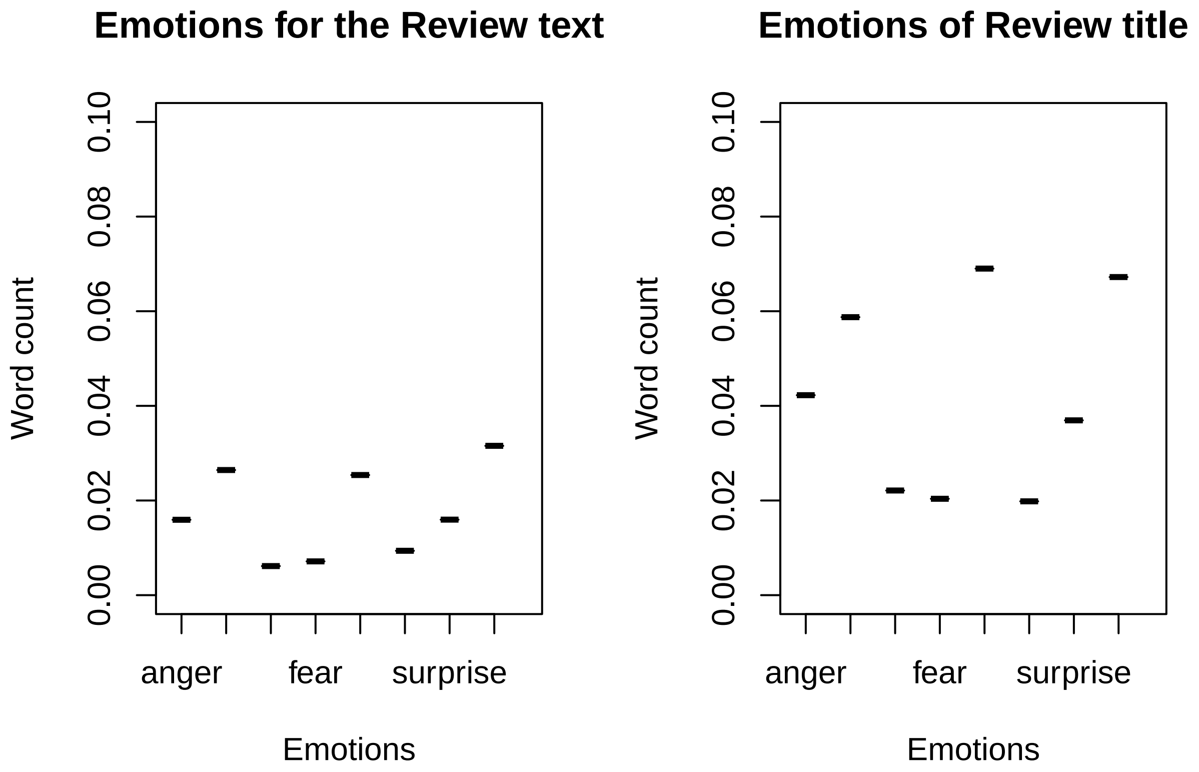


The correlation matrix provides us with information about how the different emotions might go hand in hand with another in the reviews we are examining. This reveals high positive correlation of anger with anticipation, joy and trust; of ancticipation with joy and trust; and of joy with trust. Negative correlations are less extreme but can be found between feat and disgust and feat and surprise.

Let us now also check if our emotions are similar between text and title:

```
title_emotions <- emotion_by(sentences_title)
emotiontitle_agg<-aggregate(title_emotions$ave_emotion ~ (title_emotions$emotion_type), FUN =

emotiontitle_agg <- subset(emotiontitle_agg,(1:dim(emotiontitle_agg)[1])%%2==1)

par(mfrow=c(1,2))
plot(emotiontext_agg,xaxt="n",ylab="Word count",xlab="Emotions",main="Emotions for the Review
      ylim=c(0,0.1))
```

```
axis(side=1,at=row.names(emotiontext_agg),labels=emotiontext_agg[c(1:8),1])
plot(emotiontitle_agg,xaxt="n",ylab="Word count",xlab="Emotions",main="Emotions of Review titl
     ylim=c(0,0.1))
axis(side=1,at=row.names(emotiontitle_agg),labels=emotiontitle_agg[c(1:8),1])
```



We can see that for anticipation and joy there are much higher values in the graph representing the emotions of the title, while trust is less prominent there. This means that the titles express disproportionate amounts of joy and anticipation.

Topic Analysis

Now that we have talked about the sentiments and emotions prevalent in the reviews of the product, we want to take a look at the content of the reviews. To do this efficiently, we will make use of the methods of Topic Analysis.

Topic Analysis refers to methods that aim to identify the different contents discussed and help us focus on those that we are really interested in. It functions via LDA (Latent Dirichlet allocation): after pre-processing, a document term matrix is created, the number of topics to differentiate between is decided upon, we check for convergence, estimate the model & can then interpret the first conclusion.

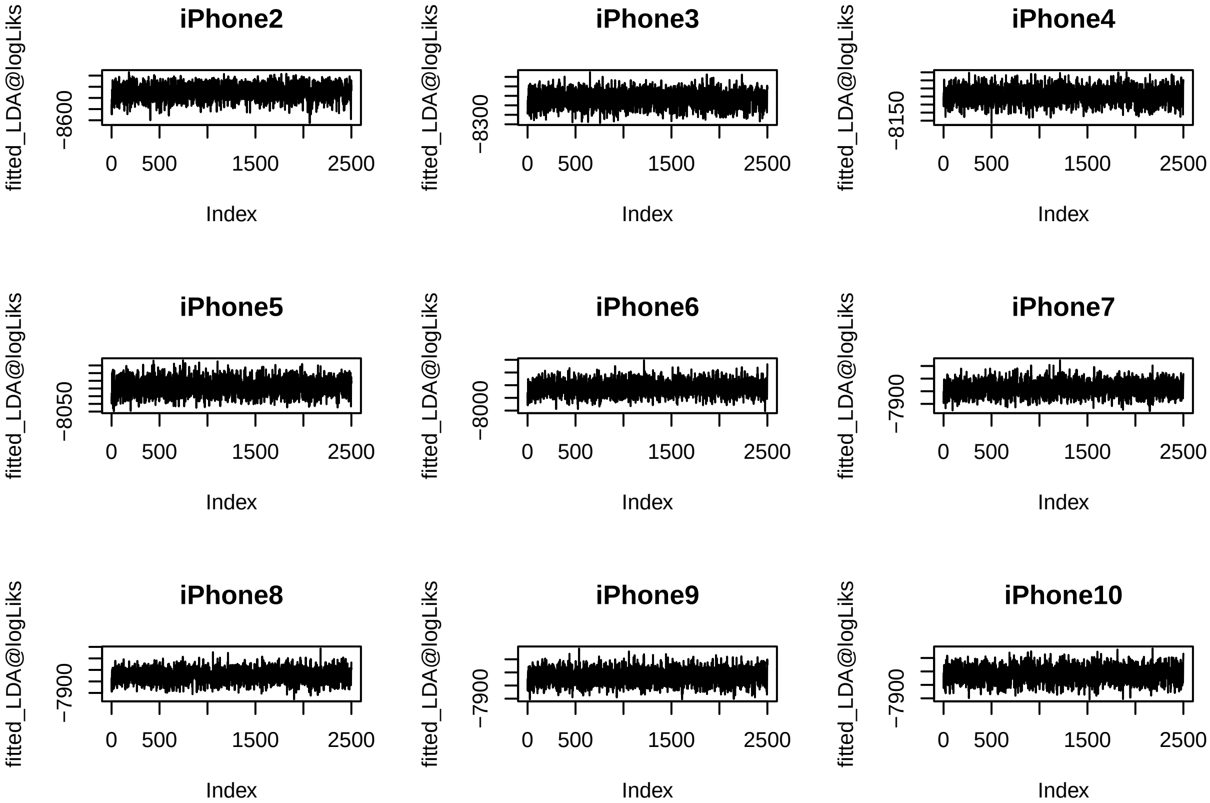
```
# already pre-processed in steps for creating the wordcloud
DocText_dtm <- DocumentTermMatrix(TextDoc)

# first steps
raw.sum<- apply(DocText_dtm,1,FUN=sum)
DocText_dtm <- DocText_dtm[raw.sum!=0,]
```

```
DocText_dtm
```

```
## <<DocumentTermMatrix (documents: 488, terms: 577)>>
## Non-/sparse entries: 1398/280178
## Sparsity           : 100%
## Maximal term length: 16
## Weighting          : term frequency (tf)
```

One of the trickiest part of topic analysis is deciding upon the right number of topics to distinguish between. To compute the optimal number, we will run the following code:

```
SEED <- 123
burnin <- 5000 #not being used for estimation
iter <- 20000  #number of iterations after burn in
keep <- 10  #keeps every tenth iteration (=burin + iter)
maxtops <- 10
avg_result_fin <- matrix(nrow=maxtops-1,ncol=3)
counter <- 1
par(mfrow=c(3,3))
for (k in 2:maxtops){

 fitted_LDA <- LDA(DocText_dtm, k = k, method = "Gibbs",
                    control = list(seed = SEED,burnin = burnin, iter = iter, keep = keep) )
 plot(fitted_LDA@logLiks,type="l", main=paste0(c("iPhone",k),collapse=""))
 words_LDA <- dim(posterior(fitted_LDA)[[1]])[2]
 avg_result_fin[counter,] <- cbind(k, logLik(fitted_LDA),-2*logLik(fitted_LDA)+(k+k*words_LDA)
 counter=counter+1
}
```



This for loop iterates and finds the best number of topics for our dataset.

```
colnames(avg_result_fin) <- c("ntopics","ll","AIC")
avg_result_fin
```

```
##       ntopics        ll      AIC
##  [1,]       2 -8421.724 17999.45
##  [2,]       3 -8105.016 17944.03
##  [3,]       4 -7897.937 18107.87
##  [4,]       5 -7826.560 18543.12
##  [5,]       6 -7634.700 18737.40
##  [6,]       7 -7663.824 19373.65
##  [7,]       8 -7635.237 19894.47
##  [8,]       9 -7640.158 20482.32
##  [9,]      10 -7568.215 20916.43
```

We make our decision on the number of topics based on AIC. We want to minimize AIC and thus choose 3 topics.

As the next step, we estimate our topic model.

```
ktop <- 3

fitted_LDA_model <- LDA(DocText_dtm, k = ktop, method = "Gibbs",
                           control = list(seed = SEED,burnin = burnin, iter = iter, keep = kee
```

From our model, we extract the topics.

```
topics <- posterior(fitted_LDA_model)[[1]]
```

Afterwards we look at the ten most important words for each topic in orther to get a grasp on what the topics are about.

```
#look at the ten most important words for the topics
for (k in 1:ktop){
  print(sort(topics[k,],decreasing=T)[1:10])
}
```

```
##       good       work      iphon       like      excel         ’s    purchas
## 0.09529907 0.08249497 0.05688678 0.02944942 0.02396195 0.01847448 0.01298701
##       issu  condition   excelent
## 0.01298701 0.01115786 0.01115786
##    batteri        buy    perfect    qualiti    speaker      iphon        use
## 0.07307465 0.03762064 0.02383297 0.01989364 0.01989364 0.01792397 0.01595430
##    scratch     defect       nice
## 0.01595430 0.01595430 0.01398464
##      great     screen        new     condit       life      worth     seller
## 0.07070707 0.04211931 0.04021346 0.03259005 0.02496665 0.01734324 0.01734324
##      price     replac        get
## 0.01353154 0.01162569 0.01162569
```

We label topic 1 as **quality**, topic 2 as **hardware** due to the high proportion of comments mentioning hardware aspects and potential issues, and topic 3 as **value**, since the product we are analysing is refurbished, and the reviews mention words like condition, worth, seller and price.

Next we create the dataframe multi, which shows the relative importance of the three topics for each review.

```
multi <- posterior(fitted_LDA_model)[[2]]
dim(multi)
```

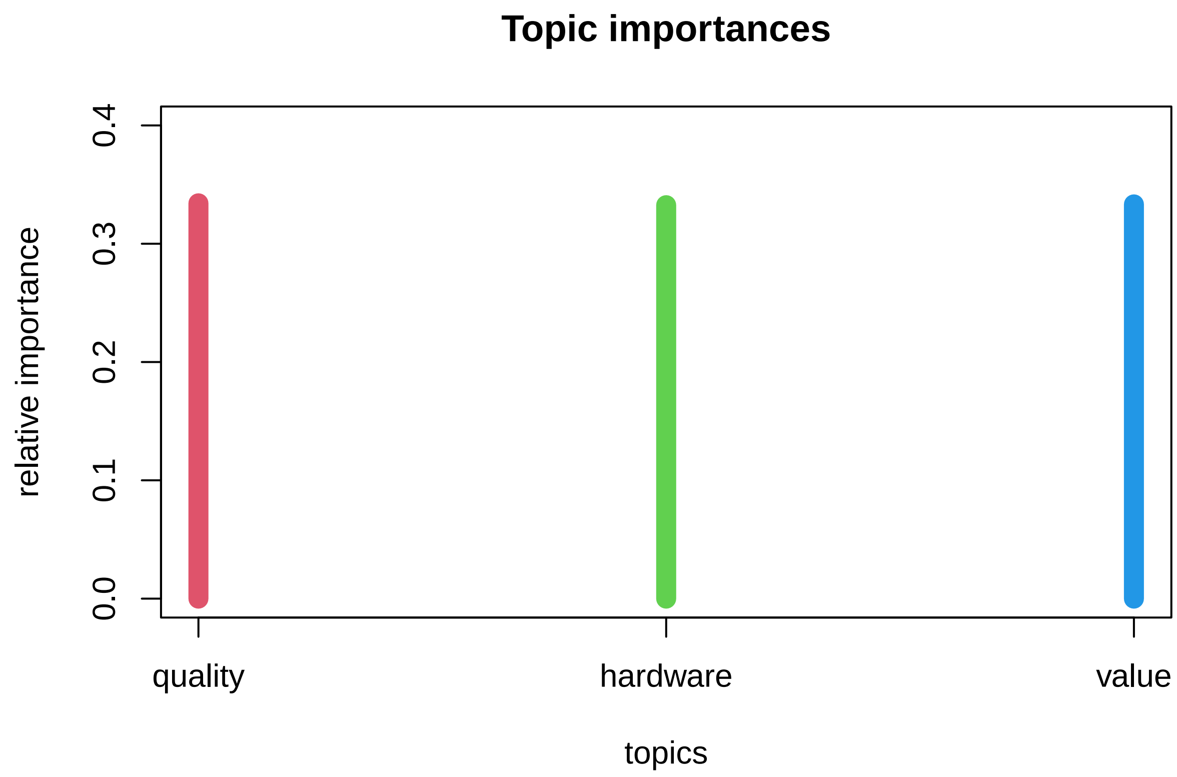
```
## [1] 488   3
```

```
colnames(multi) <- c("quality", "hardware", "value")
summary(multi)
```

```
##     quality          hardware          value       
##  Min.   :0.2825   Min.   :0.2924   Min.   :0.2924  
##  1st Qu.:0.3205   1st Qu.:0.3205   1st Qu.:0.3205  
##  Median :0.3333   Median :0.3272   Median :0.3306  
##  Mean   :0.3342   Mean   :0.3326   Mean   :0.3333  
##  3rd Qu.:0.3457   3rd Qu.:0.3397   3rd Qu.:0.3397  
##  Max.   :0.3827   Max.   :0.3908   Max.   :0.3869  
```

We plot the average imortance of the topics.

```
plot(colSums(multi)/dim(multi)[1],type="h", main="Topic importances",ylim=c(0,.4),ylab="relati
axis(side=1,at=1:3,labels=colnames(multi))
```



From the plot above we can observe that the relative importance per review of each topic is about the same (a third), meaning the three topics are of equal importance in the reviews. Looking back on the summary statistics above, we can also see that the minimum importances of all topics are just under 0.3, and the maxima for all three are below 0.4, which implies that all of our reviews include all three topics relatively evenly.

It is worth to note however, that since we are working with a limited amount of data (488 reviews), it can easily be the case that we simply do not have enough data to clearly identify topics.

Modeling approaches

In the following we will attempt to model the ratings of our product using different features that we have derived in the previous chapters.

Modeling ratings based on **date**:

```
stars_1 <- lm(star_rating_num ~  as.factor(strftime(formatted_dates, "%m")), data=reviews_scrap
summary(stars_1)
```

```
##
## Call:
## lm(formula = star_rating_num ~ as.factor(strftime(formatted_dates,
##     "%m")), data = reviews_scraped)
##
## Residuals:
##    Min     1Q Median     3Q    Max
## -2.353 -1.017  0.000  1.175  2.000
##
```

```
## Coefficients:
##                                             Estimate Std. Error t value
## (Intercept)                                  2.28571    0.52142   4.384
## as.factor(strftime(formatted_dates, "%m"))02  0.01429    0.67985   0.021
## as.factor(strftime(formatted_dates, "%m"))03  0.07792    0.66701   0.117
## as.factor(strftime(formatted_dates, "%m"))04 -0.28571    0.71399  -0.400
## as.factor(strftime(formatted_dates, "%m"))05 -0.28571    0.61954  -0.461
## as.factor(strftime(formatted_dates, "%m"))06  0.28571    0.56320   0.507
## as.factor(strftime(formatted_dates, "%m"))07  0.53968    0.54963   0.982
## as.factor(strftime(formatted_dates, "%m"))08  0.73123    0.55149   1.326
## as.factor(strftime(formatted_dates, "%m"))09  0.78836    0.55419   1.423
## as.factor(strftime(formatted_dates, "%m"))10  1.06723    0.53467   1.996
## as.factor(strftime(formatted_dates, "%m"))11  0.96148    0.54154   1.775
## as.factor(strftime(formatted_dates, "%m"))12  0.21429    0.86468   0.248
##                                             Pr(>|t|)    
## (Intercept)                                 1.43e-05 ***
## as.factor(strftime(formatted_dates, "%m"))02   0.9832    
## as.factor(strftime(formatted_dates, "%m"))03   0.9070    
## as.factor(strftime(formatted_dates, "%m"))04   0.6892    
## as.factor(strftime(formatted_dates, "%m"))05   0.6449    
## as.factor(strftime(formatted_dates, "%m"))06   0.6122    
## as.factor(strftime(formatted_dates, "%m"))07   0.3266    
## as.factor(strftime(formatted_dates, "%m"))08   0.1855    
## as.factor(strftime(formatted_dates, "%m"))09   0.1555    
## as.factor(strftime(formatted_dates, "%m"))10   0.0465 *  
## as.factor(strftime(formatted_dates, "%m"))11   0.0764 .  
## as.factor(strftime(formatted_dates, "%m"))12   0.8044    
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.38 on 488 degrees of freedom
## Multiple R-squared:  0.07125,    Adjusted R-squared:  0.05032 
## F-statistic: 3.404 on 11 and 488 DF,  p-value: 0.000145
```

This model takes a look at the star ratings as a function of time. We use the months as factor variables with 02 - February representing our baseline. However, we find that no date-parameter has a sufficiently low p-value to be considered significant.

Modeling ratings based on **review length**:

```
stars_2 <- lm(star_rating_num ~ review_length, data=reviews_scraped)
summary(stars_2)
```

```
##
## Call:
## lm(formula = star_rating_num ~ review_length, data = reviews_scraped)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -2.2663 -1.1481 -0.1435  0.9823  3.8386
##
## Coefficients:
##                Estimate Std. Error t value Pr(>|t|)
## (Intercept)    3.4319674  0.0807214   42.52  < 2e-16 ***
## review_length -0.0061366  0.0007719   -7.95 1.26e-14 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.335 on 498 degrees of freedom
## Multiple R-squared:  0.1126, Adjusted R-squared:  0.1108
## F-statistic: 63.21 on 1 and 498 DF,  p-value: 1.257e-14
```

This model takes a look at how the review length impacts the star ratings in closer detail. We find out that actually, the review length does have a significant impact in determining the stars given in a review. This relationship is negative, meaning: the longer a review, the lower the average star rating.

Modeling ratings based on **sentiment**:

```
stars_3 <- lm((star_rating_num) ~ (sentiment_title$ave_sentiment) + (sentiment_text$ave_sentime
              + (review_length)
               ,    data=reviews_scraped)
summary(stars_3)
```

```
##
## Call:
## lm(formula = (star_rating_num) ~ (sentiment_title$ave_sentiment) +
##     (sentiment_text$ave_sentiment) + (review_length), data = reviews_scraped)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -2.62866 -0.79767  0.02889  0.79394  2.85312
##
## Coefficients:
##                                 Estimate Std. Error t value Pr(>|t|)
## (Intercept)                    3.0229077  0.0682189  44.312  < 2e-16 ***
## sentiment_title$ave_sentiment  1.3429874  0.1253157  10.717  < 2e-16 ***
## sentiment_text$ave_sentiment   1.6706981  0.2006733   8.325 8.23e-16 ***
## review_length                 -0.0041647  0.0006127  -6.797 3.08e-11 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.042 on 496 degrees of freedom
## Multiple R-squared:  0.4613, Adjusted R-squared:  0.458
## F-statistic: 141.6 on 3 and 496 DF,  p-value: < 2.2e-16
```

This model now includes the sentiment of the title of the review and of the review itself as well as the length of the review as determining parameters for the expected star rating. In the case of this model, the sentiment of title as well as the review length are considered to be significant. While a higher sentiment of the title will lead to an increase in the predicted star rating, a higher word count of the review will lead to a decrease in the predicted star rating. We also tested for a model only using sentiment, not review length - here too, only the sentiment of the title resulted in being statistically significant at alpha=0.05.

Modeling ratings based on **emotions**:

```
stars_4 <- lm((star_rating_num ~ ave_emotion.anger+ave_emotion.anticipation + ave_emotion.disgu
              ,    data=reviews_scraped)
summary(stars_4)
```

```
##
## Call:
## lm(formula = (star_rating_num ~ ave_emotion.anger + ave_emotion.anticipation +
##     ave_emotion.disgust + ave_emotion.fear + ave_emotion.joy +
##     ave_emotion.sadness + ave_emotion.surprise + ave_emotion.trust +
##     (review_length)), data = reviews_scraped)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -2.6921 -1.0501 -0.1458  0.9583  3.5966
##
## Coefficients:
##                            Estimate Std. Error t value Pr(>|t|)
## (Intercept)               3.436e+00  9.518e-02  36.097  < 2e-16 ***
## ave_emotion.anger         1.799e+00  2.394e+00   0.751  0.45287
## ave_emotion.anticipation -3.487e+00  2.246e+00  -1.553  0.12110
## ave_emotion.disgust      -1.412e+01  5.279e+00  -2.675  0.00772 **
## ave_emotion.fear         -5.898e+00  5.395e+00  -1.093  0.27483
## ave_emotion.joy           8.135e+00  2.758e+00   2.950  0.00333 **
## ave_emotion.sadness      -6.303e+00  5.643e+00  -1.117  0.26460
## ave_emotion.surprise     -3.240e+00  2.161e+00  -1.499  0.13448
## ave_emotion.trust         1.135e+00  2.017e+00   0.563  0.57365
## review_length            -5.323e-03  7.525e-04  -7.074 5.23e-12 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.276 on 490 degrees of freedom
## Multiple R-squared:  0.2018, Adjusted R-squared:  0.1871
## F-statistic: 13.76 on 9 and 490 DF,  p-value: < 2.2e-16
```

This model reveals that stars are only significantly impacted by review length, not however by any of the emotion categories.

Modeling ratings based on **sentiments & emotion**:

```
stars_5 <- lm((star_rating_num ~ ave_emotion.anger+ave_emotion.anticipation + ave_emotion.disgu
              ,    data=reviews_scraped)
summary(stars_5)
```

```
##
```

```
## Call:
## lm(formula = (star_rating_num ~ ave_emotion.anger + ave_emotion.anticipation +
##     ave_emotion.disgust + ave_emotion.fear + ave_emotion.joy +
##     ave_emotion.sadness + ave_emotion.surprise + ave_emotion.trust +
##     (sentiment_title$ave_sentiment) + (sentiment_text$ave_sentiment) +
##     review_length), data = reviews_scraped)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -2.75800 -0.77881  0.01105  0.75331  2.89580
##
## Coefficients:
##                                 Estimate Std. Error t value Pr(>|t|)
## (Intercept)                    3.0638350  0.0827222  37.038  < 2e-16 ***
## ave_emotion.anger              1.0415091  1.9524080   0.533    0.594
## ave_emotion.anticipation      -2.8119720  1.8365760  -1.531    0.126
## ave_emotion.disgust           -0.5169613  4.3975422  -0.118    0.906
## ave_emotion.fear              -4.0902856  4.4024474  -0.929    0.353
## ave_emotion.joy                2.5449571  2.2767434   1.118    0.264
## ave_emotion.sadness           -2.9618250  4.6260288  -0.640    0.522
## ave_emotion.surprise          -1.3880474  1.7661439  -0.786    0.432
## ave_emotion.trust              1.1092350  1.6438798   0.675    0.500
## sentiment_title$ave_sentiment  1.3190432  0.1262619  10.447  < 2e-16 ***
## sentiment_text$ave_sentiment   1.5839373  0.2234456   7.089 4.77e-12 ***
## review_length                 -0.0040125  0.0006195  -6.477 2.28e-10 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.04 on 488 degrees of freedom
## Multiple R-squared:  0.4717, Adjusted R-squared:  0.4598
## F-statistic: 39.61 on 11 and 488 DF,  p-value: < 2.2e-16
```

When modeling star ratings based on emotions, sentiments & review_length we find out that the emotion surprise, the sentiment of the title and the word count of the review have a significant impact in determining the final rating. Surprise and long reviews go hand in hand with lower ratings, while positive title sentiment is associated with higher ratings. The other emotion categories remain insignificant.

We can try a full model for determining if the rating will have 5 stars:

```
stars_6 <- glm((star_rating_num) > 4 ~ (ave_emotion.anger+ave_emotion.anticipation + ave_emoti
              ,    data=reviews_scraped, family = "binomial")
summary(stars_6)
```

```
##
## Call:
## glm(formula = (star_rating_num) > 4 ~ (ave_emotion.anger + ave_emotion.anticipation +
##     ave_emotion.disgust + ave_emotion.fear + ave_emotion.joy +
##     ave_emotion.sadness + ave_emotion.surprise + ave_emotion.trust +
##     sentiment_title$ave_sentiment + sentiment_text$ave_sentiment +
##     review_length), family = "binomial", data = reviews_scraped)
##
## Coefficients:
##                                Estimate Std. Error z value Pr(>|z|)
## (Intercept)                   -1.829541   0.270671  -6.759 1.39e-11 ***
## ave_emotion.anger            -10.764945   6.621895  -1.626   0.1040
## ave_emotion.anticipation      -4.088539   5.248829  -0.779   0.4360
## ave_emotion.disgust           -7.646071  18.940134  -0.404   0.6864
## ave_emotion.fear             -30.611894  20.771997  -1.474   0.1406
## ave_emotion.joy               10.428342   6.727616   1.550   0.1211
## ave_emotion.sadness            7.279685  17.188276   0.424   0.6719
## ave_emotion.surprise         -14.172392   7.091098  -1.999   0.0456 *
## ave_emotion.trust              5.336966   3.703566   1.441   0.1496
## sentiment_title$ave_sentiment  2.850839   0.425870   6.694 2.17e-11 ***
## sentiment_text$ave_sentiment   1.928125   0.595520   3.238   0.0012 **
## review_length                 -0.006096   0.002657  -2.294   0.0218 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 500.40  on 499  degrees of freedom
## Residual deviance: 333.02  on 488  degrees of freedom
## AIC: 357.02
##
## Number of Fisher Scoring iterations: 6
```

When combining all features in a model, only the average sentiment of the review title remains significant in determining whether the rating will be 5 stars.

Building a conclusive model for helfpulness:

```
help <- glm((N_helpful) > 10 ~ (ave_emotion.anger+ave_emotion.anticipation + ave_emotion.disgus
              ,    data=reviews_scraped, family = "binomial")
summary(help)
```

```
##
## Call:
## glm(formula = (N_helpful) > 10 ~ (ave_emotion.anger + ave_emotion.anticipation +
##     ave_emotion.disgust + ave_emotion.fear + ave_emotion.joy +
##     ave_emotion.sadness + ave_emotion.surprise + ave_emotion.trust +
##     sentiment_title$ave_sentiment + sentiment_text$ave_sentiment +
##     review_length), family = "binomial", data = reviews_scraped)
##
## Coefficients:
##                                Estimate Std. Error z value Pr(>|z|)
## (Intercept)                   -6.269214   0.830195  -7.551 4.30e-14 ***
## ave_emotion.anger             14.943941  14.002050   1.067    0.286
## ave_emotion.anticipation       -7.055534  24.519837  -0.288    0.774
## ave_emotion.disgust           -42.019103  36.631222  -1.147    0.251
## ave_emotion.fear                9.403853  32.835695   0.286    0.775
## ave_emotion.joy                 1.881173  26.137013   0.072    0.943
## ave_emotion.sadness            42.026397  31.926990   1.316    0.188
## ave_emotion.surprise            4.319313  31.649544   0.136    0.891
## ave_emotion.trust             -15.913280  23.175217  -0.687    0.492
## sentiment_title$ave_sentiment   0.752852   0.747336   1.007    0.314
## sentiment_text$ave_sentiment    1.559535   1.757458   0.887    0.375
## review_length                   0.022442   0.003112   7.212 5.51e-13 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 215.82  on 499  degrees of freedom
## Residual deviance: 108.97  on 488  degrees of freedom
## AIC: 132.97
##
## Number of Fisher Scoring iterations: 8
```

This model to determine if a review will be voted as helpful by more than 10 people shows that only the review_length, not the emotion or sentiment of the review is significant in shaping the outcome.